From Recovery to Locomotion: Learning Robust Humanoid Control via Curriculum and Policy Distillation

Stefanie Theodora Karolina Gschwind

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2025-87
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2025/EECS-2025-87.html

May 16, 2025

From Recovery to Locomotion: Learning Robust Humanoid Control via Curriculum and Policy Distillation

by

Author Stefanie Theodora Karolina Gschwind

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.
Approval for the Report and Comprehensive Examination:

Committee:

Professor Sergey Levine
Research Advisor

May 14, 2025

(Date)

Professor Pieter Abbeel
Second Reader

(Date)

From Recovery to Locomotion: Learning Robust Humanoid Control via Curriculum and Policy Distillation

by

Stefanie Theodora Karolina Gschwind

A thesis submitted in partial satisfaction of the

requirements for the degree of

Masters of Science

in

Electrical Engineering & Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Sergey Levine, Chair
Professor Pieter Abbeel

Spring 2025

From Recovery to Locomotion: Learning Robust Humanoid Control via Curriculum and Policy Distillation

Abstract

From Recovery to Locomotion: Learning Robust Humanoid Control via Curriculum and Policy Distillation

by

Stefanie Theodora Karolina Gschwind

Masters of Science in Electrical Engineering & Computer Science

University of California, Berkeley

Professor Sergey Levine, Chair

Humanoid robots must operate robustly in dynamic, unstructured environments where falling is inevitable and manual resets are infeasible. While prior work has achieved impressive locomotion or recovery performance in isolation, few approaches address the challenge of unified, full-body control in real-world conditions. This thesis introduces an end-to-end framework for generalizable, robust control that enables humanoid robots to walk, fall, recover, and resume walking autonomously.

We first extend morphology-randomized training from quadrupeds to bipeds and humanoids, uncovering critical limitations in gait stability and failure recovery when generalizing across designs. To overcome these limitations, we propose a keyframe-based recovery curriculum that decomposes the complex getup task into human-inspired phases—transitioning through a kneeling intermediate pose to improve energy efficiency, stability, and success rate. We then integrate recovery and locomotion policies into a single unified controller using Dataset Aggregation (DAgger), enabling seamless transitions between walking and getting up without brittle switching logic.

Evaluated on realistic simulated environments featuring low obstacles, slippery patches, and external perturbations, our unified policy demonstrates superior robustness and consistency compared to baseline methods. This work provides a deployable control architecture for humanoid robots that bridges the gap between simulation and real-world demands, laying a foundation for lifelong, autonomous operation.

To my parents, sister, and friends

Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I would like to extend my deepest and most heartfelt thanks to Professor Sergey Levine. His brilliance, generosity, and vision have left a lasting impact on me, and it has been an incredible privilege to conduct research under his guidance. His mentorship has shaped not only this thesis, but also my growth as a researcher. I am endlessly grateful for the opportunity to learn from him.

I am especially grateful to Laura Smith, who has been my closest research mentor throughout my time at UC Berkeley. Her unwavering support, patience, and insight have profoundly shaped my research experience. Her mentorship has been the foundation of nearly every project I've pursued, and her belief in me has meant more than I can express.

I would also like to sincerely thank Kevin Zakka, whose thoughtful feedback and steady guidance throughout this thesis have been invaluable. Working alongside Laura, Kevin helped bring clarity and direction to my work at every stage, and I'm very grateful for his support.

I am also thankful to Xue Bin Peng and Zhongyu Li for their mentorship and collaboration on earlier work that is referenced in this thesis.

This thesis—and the research journey behind it—would not have been possible without the support of Professor Sergey Levine and Laura Smith. I am truly and deeply grateful.

Chapter 1

Introduction

Recent advancements in artificial intelligence (AI), particularly in large-scale language models such as ChatGPT, have catalyzed a reimagining of what general intelligence might look like in robotics. While AI systems have shown remarkable versatility in domains like language, reasoning, and decision-making, robotic systems still lag behind in terms of general-purpose capability. The success of foundation models has sparked interest in building more adaptable robotic agents that can operate autonomously in unstructured, dynamic environments. Yet, realizing this vision requires moving beyond rigid, task-specific robotic systems to develop controllers that are robust, generalizable, and capable of lifelong adaptation.

Historically, robotic control systems have been designed for narrowly defined applications, often in highly structured environments. Industrial robots, for instance, excel at executing precise, repetitive tasks on factory floors, where their motion and surroundings are carefully constrained. However, real-world deployment—from homes and hospitals to disaster zones and outdoor terrain—demands much more flexible systems. Robots must navigate diverse surfaces, respond to unpredictable perturbations, and recover gracefully from failure. This necessitates the development of locomotion and whole-body control strategies that can generalize across scenarios and platforms.

1.1 The Case for Legged Robots

Legged robots offer a compelling foundation for general-purpose autonomy. Unlike wheeled platforms, which are confined to flat and obstacle-free environments, legged systems can traverse complex, uneven terrain. Their morphology enables them to adapt their posture and reposition their bodies to interact with objects that are out of reach or obstructed. These capabilities make legged robots especially suitable for tasks in cluttered, unpredictable settings such as homes, construction sites, or search-and-rescue missions.

In addition to their mobility, legged robots offer a larger effective workspace for manipulation. Whereas mobile manipulators based on wheeled bases or stationary arms are constrained in their reach and terrain adaptability, a legged robot can crouch, climb, or

reposition itself to extend its interaction range. Moreover, their ability to recover from falls—a key capability for long-term autonomy—is uniquely tied to their morphology and full-body coordination.

1.2 Motivation for Robust and Safe Recovery

In real-world deployments, humanoid robots are inevitably going to fail. Whether due to uneven terrain, unexpected contact, or internal malfunctions, falls are an unavoidable part of long-term autonomous operation. Thus, an effective robot must possess a reliable reset policy—a way to recover from fallen states that is consistent, safe, and efficient.

Many existing recovery policies, especially those deployed in real-world scenarios, are scripted by hand. While these may work in constrained settings, they offer no guarantees of generalizing to new environments with different terrain properties, contact dynamics, or initial conditions. Furthermore, many learned recovery behaviors tend to produce erratic, high-energy motions that result in repeated failures, unnecessary stress on hardware, and potentially dangerous interactions with the environment.

Our goal is to develop a reset controller that avoids these issues: it should conserve energy, minimize flailing, and succeed with as few attempts as possible. In other words, recovery should be stable, repeatable, and physically realistic.

To achieve this, we draw inspiration from human movement. When people fall, they typically adopt postures like kneeling or pushing from all fours to return to standing—strategies that minimize energy expenditure and avoid risky joint configurations. We model our recovery controller with a keyframe-based curriculum centered on such human-like motions, using an intermediate kneeling pose as a transition point between fallen and standing states. This approach leads to safer and more efficient recovery trajectories that better align with the constraints of real-world deployment.

1.3 Research Objectives

This thesis aims to develop robust, generalizable controllers for legged robots using reinforcement learning (RL). The central focus is on enabling full-body recovery and locomotion capabilities for humanoid robots—two critical skills for real-world deployment. We explore this through a series of contributions:

- We extend morphology-randomized locomotion training from GenLoco to the bipedal and humanoid regime, evaluating the limits of cross-morphology generalization.
- We propose a curriculum-based recovery policy trained with keyframe decomposition, enabling safe and reliable getup behaviors from diverse fall configurations.
- We design a unified policy that integrates recovery and locomotion behaviors through teacher-student distillation and online Dataset Aggregation (DAgger).

- We evaluate our unified controller on challenging environments that simulate real-world obstacles, including low walls, slippery surfaces, and physical perturbations.

Through these contributions, we demonstrate that modular decomposition combined with end-to-end training can produce policies that are both robust and generalizable—a critical step toward real-world-ready humanoid robots.

1.4 Thesis Structure

The remainder of this thesis is organized as follows:

- Chapter 2 provides background on reinforcement learning and prior work in locomotion and recovery.
- Chapter 3 describes our efforts to extend GenLoco-style morphology generalization to bipedal and humanoid platforms, and the insights gained from this process.
- Chapter 4 presents our recovery controller, detailing its curriculum design, reward structure, and comparison to baseline methods.
- Chapter 5 introduces our skill integration framework using teacher-student learning and DAgger to produce a unified locomotion and recovery policy.
- Chapter 6 evaluates the resulting controller across multiple real-world-inspired environments.
- Chapter 7 concludes with a summary of findings and directions for future research.

Contributions of This Work

Chapter 3 was completed in collaboration with Vishnu Sangli, under the guidance of Xue Bin (Jason) Peng, Zhongyu Li, and Laura Smith. This work reflects a joint effort in implementing and evaluating cross-morphology locomotion controllers.

The remainder of the thesis—spanning curriculum design, locomotion-recovery integration, training pipeline development, evaluation, and writing—was conducted independently by the author, with technical feedback and mentorship from Laura Smith and Kevin Zakka.

Chapter 2

Background

Reinforcement Learning for Robotic Control

Reinforcement learning (RL) has emerged as a powerful paradigm for robotic control, particularly in continuous control tasks where dynamics are complex or difficult to model analytically. In model-free RL, agents learn to maximize a reward signal by directly interacting with an environment, bypassing the need for an explicit dynamics model. Common algorithms like Proximal Policy Optimization (PPO) [60] and Soft Actor-Critic (SAC) [28] have demonstrated strong performance on tasks ranging from manipulation to locomotion [27, 52, 48, 32, 64, 66, 5, 1].

In robotic locomotion, the design of the reward function plays a central role in shaping behavior [53, 29, 68, 16, 2, 15, 42, 67]. Sparse or poorly aligned rewards can lead to unstable or suboptimal policies, making curriculum learning strategies especially valuable for progressively shaping complex skills such as standing, walking, and turning [4, 21, 20, 6]. RL has been successfully used to train robust controllers for quadrupeds and humanoids, including recent work by Smith et al., who apply RL to learn energy-efficient walking and recovery motions [64, 61, 63, 62].

Despite these successes, RL-based methods still face several significant challenges. Chief among them is poor sample efficiency, with modern algorithms often requiring millions of environment interactions to converge [1, 34]. Additionally, policies trained exclusively in simulation frequently fail to transfer effectively to the real world due to discrepancies in system dynamics, contact modeling, and sensor characteristics—an issue commonly referred to as the sim-to-real gap [69, 74]. A wide body of work has sought to address this challenge through strategies such as randomized dynamics during training, modeling actuator delays, and injecting observation noise to promote robustness [69, 3, 36, 12]. Nevertheless, additional issues such as mode collapse—where policies exhibit unstable or repetitive behaviors—and real-world safety remain critical barriers to deployment.

Learned Locomotion Policies

The development of locomotion controllers has historically been constrained to specific morphologies, with much early work targeting quadrupeds or highly simplified humanoids [33, 66, 14, 41]. With advances in simulation and computational resources, researchers have begun exploring morphology-randomized training to achieve cross-design generalization [44, 17, 33]. For example, the GenLoco framework demonstrated that a single policy could generalize across a family of quadruped and hexapod morphologies [17], though extensions to bipedal and humanoid forms remain less developed [45, 33] due to challenges in balance and contact planning.

Humanoids, in particular, pose unique difficulties due to their underactuation, high center of mass, and complex foot-ground interactions [25, 26, 39]. While terrain randomization and domain randomization strategies have enabled some degree of robustness [69], the development of controllers that can recover from significant disturbances—such as falls—remains under-explored [23].

Whole-Body and Recovery Control

In real-world deployments, robots must be capable not only of walking, but also of recovering from falls or unstable configurations. Fall recovery is especially important for humanoid robots, which are prone to tipping due to narrow support polygons and high centers of mass [22].

Traditional recovery controllers rely on either motion planning or replaying hand-designed trajectories [65, 24, 14]. For instance, many commercial robots include built-in getup routines that are manually tuned for specific fall poses [40, 71]. However, these approaches lack generalization and can fail when the robot falls into unexpected configurations.

Recent research has explored RL-based approaches to fall recovery. Inspired by character animation techniques like DeepMimic [56] and Adversarial Motion Priors (AMP) [54], these methods learn to imitate getup motions or optimize recovery behaviors directly [18, 75]. While successful in simulation, such methods often rely on high-frequency torque profiles and unrealistic dynamics models, making them difficult to transfer to real hardware [7]. Others have adopted hierarchical or hybrid control [11, 35, 24], combining scripted motion with reactive planning to improve success rates.

A promising direction involves using keyframe-based motion decomposition to simplify the learning problem. By defining a small number of physically interpretable key poses and training policies to interpolate or transition between them, these controllers achieve more stable and reproducible behavior [23, 65]. This structure not only aids learning but also improves physical plausibility—a critical factor for real-world execution.

Multi-Skill Integration and Policy Distillation

Robust control in unstructured environments often requires the integration of multiple skills, such as walking and recovering from a fall [72, 57]. Naively switching between policies can cause instability or unsafe transitions. As a solution, recent work has proposed multi-skill architectures that leverage hierarchical or multi-modal policies [19, 55, 50].

In particular, the Dataset Aggregation (DAgger) framework offers a principled way to train a unified student policy that distills expert demonstrations—including those from modular controllers or scripted primitives—into a single network [46, 58, 59, 48, 31]. Originally proposed to address distributional drift in imitation learning, DAgger has found widespread adoption in robotics for integrating diverse behaviors, such as locomotion and manipulation or walking and recovery [49, 9, 8, 10, 37, 51, 43, 30].

In this thesis, we apply DAgger to integrate keyframe-based recovery with joystick-based locomotion into a single end-to-end controller. This approach addresses both the robustness and modularity required for real-world deployment.

Simulation Tools and Hardware Platforms

The choice of simulation platform plays a critical role in the development and evaluation of robotic controllers. MuJoCo [70] and its JAX-based derivative MJX [73] provide fast, differentiable simulation capabilities with accurate contact dynamics, making them ideal for RL-based training. Compared to other platforms such as PyBullet [13] or IsaacGym [47], MJX offers better integration with gradient-based learning libraries and improved simulation fidelity [66, 38].

This work uses the Booster T1 humanoid robot as the target platform. The T1 represents a class of low-cost humanoid robots that are both accessible and mechanically capable, offering a compelling platform for testing general-purpose controllers. Unlike highly specialized or over-actuated systems, the T1's minimalistic design demands controllers that are both energy-efficient and resilient.

Taken together, the integration of simulation fidelity, policy structure, and training methodology in this work advances the field toward generalizable and robust full-body control for humanoid robots.

From Prior Work to This Thesis: Contributions in Context

Prior approaches to humanoid locomotion and recovery have demonstrated strong results within constrained scopes—whether through hand-designed controllers, motion planning with restricted initializations, or reinforcement learning applied in narrowly defined state spaces. Model-based techniques such as ZMP control or trajectory optimization have yielded impressive locomotion and manipulation capabilities, but often fall short in unstructured, contact-rich settings like fall recovery. Meanwhile, learned controllers, including DeepMimic-style imitation or hierarchical RL approaches, have shown promise in simulation but fre-

quently lack the physical realism, transferability, or modularity needed for deployment on real-world humanoid hardware.

In this thesis, we introduce a novel end-to-end framework that addresses several open challenges in full-body humanoid control by combining structured motion priors with modern reinforcement learning:

- **Keyframe-Based Recovery Curriculum:** We introduce a decomposition of the recovery problem using human-inspired intermediate poses. Unlike prior works that either hard-code fall recovery motions or train from flat reward functions, our approach uses kneeling as an intermediate pose and optimizes transitions from fallen to kneeling, and kneeling to standing via reinforcement learning. This improves robustness and energy efficiency across diverse initial conditions.

- **Unified Locomotion-Recovery Integration:** We propose a modular teacher-student framework that combines robust locomotion and recovery behaviors into a single policy using online Dataset Aggregation (DAgger). This avoids the pitfalls of brittle policy switching or multi-headed networks, allowing seamless and interpretable skill blending.

- **Realistic Evaluation Environments:** We construct challenging, physically grounded test environments—including low walls, puddles, and external perturbations—to probe generalization and deployment-readiness. Unlike works that evaluate solely in controlled settings, our evaluations emphasize consistency, physical plausibility, and spatial safety.

- **Quantitative and Qualitative Gains:** Our evaluations reveal that the proposed controller not only reduces energy and joint stress during getup, but also exhibits emergent corrective behaviors (e.g., rebalancing mid-slip or avoiding repeated failed getup attempts). These insights point toward an internalized notion of stability learned from curriculum and imitation, beyond what scripted or end-to-end RL policies typically exhibit.

Together, these contributions push beyond the boundaries of prior research by proposing a controller that is not only robust in simulation, but grounded in physical realism and designed with deployment in mind. This work lays the foundation for future humanoid systems that can walk, fall, and recover—autonomously, adaptively, and safely.

Chapter 3

Extending GenLoco to Bipeds: Challenges in Morphology Generalization

3.1 Introduction

Inspired by the success of the original GenLoco work in developing a single, robust locomotion controller capable of generalizing across a wide range of quadruped morphologies, we set out to explore whether the same approach could be extended to more challenging robot designs—namely bipeds and humanoids. Our goal was to evaluate the limits of GenLoco-style multitask training when applied to extreme morphology transfer, and to investigate whether a single controller could learn to walk across both quadrupedal and bipedal platforms with vastly different kinematics and dynamic constraints.

This investigation was directly motivated by the broader objective of building robust, real-world-capable humanoid systems. Unlike quadrupeds, humanoid robots are more likely to be deployed in human-centric environments and thus must recover from failure gracefully and safely. However, our initial hypothesis was that generalization across morphologies would provide a path to controllers that are inherently adaptable and robust. As we discovered, however, morphology generalization alone is insufficient for real-world robustness—particularly when recovery from falls is a requirement. These limitations revealed a critical gap in prior work and motivated a shift toward developing task-decomposed, modular policies targeted at failure recovery and resilience in a single morphology.

3.2 Experimental Setup

Simulation

All experiments were conducted in NVIDIA's IsaacGym simulator, chosen for its GPU-accelerated RL training pipeline. IsaacGym supports fully parallelized physics simulation and policy training directly on the GPU, making it ideal for the large-scale data requirements of our study. In practice, training each stage of the experiment still required 24–48 hours of wall-clock time, underscoring the complexity of learning across diverse morphologies.

To enable simultaneous training on multiple robots, we modified the IsaacGym backend to support agents with different action and observation spaces in the same environment—a key requirement for our multitask learning framework. While IsaacGym facilitated faster training, we encountered several challenges: poor documentation, lack of support from NVIDIA, and broken multi-GPU support, which limited scalability despite access to multiple GPUs.

Robots

We selected five robots to serve as representatives across the quadruped-biped spectrum:

- **Quadrupeds**: Unitree A1, MIT Mini Cheetah
- **Bipeds/Humanoids**: Cassie (leg-only), Berkeley Humanoid (torso + legs), Unitree H1 (full humanoid with arms)

Our selection focused on maximizing morphological diversity among bipeds while keeping quadrupeds limited (as GenLoco had already demonstrated generalization among them). These choices allowed us to explore transfer learning not only between robots of different morphologies, but also different sizes and complexity levels—e.g., the H1 stands twice as tall as the Berkeley Humanoid. We also strategically selected platforms that were physically available to us for potential real-world transfer, and those our lab had prior experience with, enabling smoother troubleshooting and debugging.

Morphology Randomization

Building on GenLoco's approach, we extended morphology randomization to bipedal robots for the first time. To reduce complexity, we created simplified versions of each robot that preserved kinematic and dynamic features critical for locomotion. Our randomization pipeline generated thousands of physically feasible variants by varying:

- Limb dimensions (leg/arm lengths, joint widths)
- Mass distribution (torso and limb mass, CoM location)
- Base structure (torso length, width)

- Ground contact points (foot area and geometry)

We tuned the randomization ranges manually through reasoning and iterative testing, ensuring each generated morphology remained within simulation constraints. Collision models were updated on the fly to reflect structural changes, avoiding self-collisions and infeasible behaviors. This approach enabled us to simulate robot variation akin to real-world deviations due to manufacturing tolerances or mechanical wear.

3.3 Training Strategy

Training was structured in a curriculum-inspired sequence to mitigate instability and catastrophic failures:

- **Single-robot training**: Basic walking policy for each original morphology.
- **Intra-family generalization**: Training across morphology-randomized variants of one robot.
- **Intra-class transfer**: Generalizing across either all quadrupeds or all bipeds.
- **Cross-class generalization**: A single policy for both bipedal and quadrupedal robots.

Attempting to train directly on all robots at once quickly led to instability, exploding gradients, and NaNs. Breaking down the problem into manageable chunks allowed us to identify appropriate reward functions and hyperparameter settings for each morphology class.

3.4 Results and Insights

This staged approach allowed us to gradually scale the complexity of the problem and to develop robust reward structures that supported diverse gaits. In the process, several critical insights emerged:

- **Quadrupeds consistently outperformed bipeds.** Even in early stages, quadruped locomotion policies were stable and robust. Biped policies, while functional in simulation, exhibited greater variance in performance.
- **Biped gait quality suffered.** Cassie-derived morphologies were prone to falling, often before the episode ended. H1-based humanoids adopted inefficient strategies like foot shuffling instead of coordinated steps.
- **Recovery capabilities diverged.** We successfully trained reset controllers for quadrupeds, allowing them to recover from falls mid-episode. In contrast, all attempts to train bipedal reset behaviors failed, likely due to the inherent instability and high-dimensional contact dynamics of bipedal recovery.

These findings formed the pivotal realization that generalization across morphologies without explicit recovery is fundamentally limited. While locomotion generalization was achievable to some extent, true robustness—in the sense of autonomy and failure resilience—remained elusive without a clear decomposition of capabilities. Moreover, as our goal was to build controllers that would be viable for real-world deployment on humanoid hardware, we recognized that generalization at the expense of safety and consistency was counterproductive. Humanoids must not only walk, but also recover in ways that are physically plausible, energy-efficient, and not erratic or unsafe.

This motivated a key shift in research focus: rather than pursuing unified general-purpose controllers across morphology classes, we turned toward a modular, end-to-end approach focused solely on humanoid robots. The rest of this thesis focuses on developing:

- A robust reset controller that enables a humanoid to autonomously recover from arbitrary fall configurations.
- A resilient walking controller capable of handling terrain variability and perturbations.
- A policy distillation framework that merges these capabilities into a unified, deployable system.

This transition marks a deliberate step away from morphology-agnostic generalization toward building truly deployable, robust controllers—with the T1 humanoid as our central testbed.

Chapter 4

Recovery Control for Autonomous Resilience

4.1 Motivation and Overview

To develop a robust and deployable reset policy for humanoid robots, we shifted our experimental focus away from morphology-agnostic generalization toward specialized training on a single platform. In contrast to locomotion, recovery from falls requires full-body dynamic coordination and safe, reliable motor usage. Reinforcement learning (RL) offers the ability to train a controller that is both robust to environmental perturbations and generalizable across fall configurations.

More importantly, our work is motivated by the reality that humanoid robots operating in real-world environments will inevitably fall. Without a reliable and physically plausible recovery mechanism, these robots will remain unsuitable for deployment in homes, workplaces, or crowded public settings. Many existing recovery strategies are either hand-scripted or learned through reward shaping alone, leading to behaviors that are erratic, overly aggressive, or highly sensitive to initial conditions. These methods often result in policies that flail uncontrollably, use excessive energy, or require multiple attempts before succeeding. Such behaviors not only stress hardware components but can also pose safety risks in cluttered environments.

In contrast, our objective is to develop a recovery controller that behaves more like a human: consistent, low-energy, and physically stable. Humans tend to get up in ways that minimize strain and risk, often transitioning through intermediate postures such as kneeling or crawling before standing. This observation inspired the design of a keyframe-based recovery strategy that mimics human recovery behavior. By structuring the recovery task into a two-stage curriculum centered around a human-inspired kneeling pose, we are able to produce smoother, more deliberate recovery behaviors that succeed reliably and safely across a wide range of fall configurations.

This chapter presents our approach to developing a robust recovery controller for the

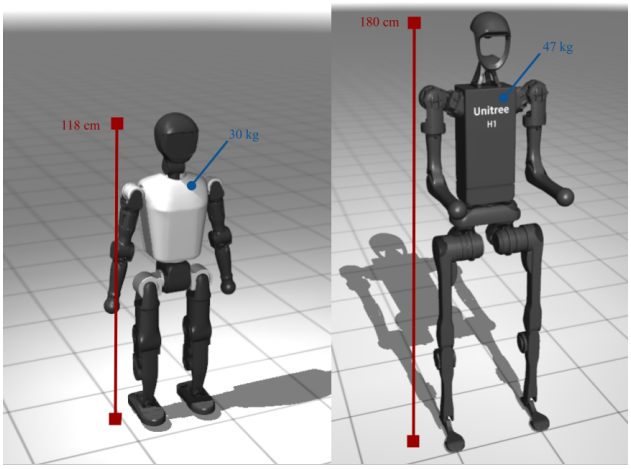

Figure 4.1: **Left:** T1 robot XML loaded into the flat MuJoCo scene. **Right:** H1 robot XML loaded into the flat MuJoCo scene.

Booster T1 humanoid. We describe our robot selection criteria, simulation environment, curriculum design, and reward formulation, all of which are geared toward ensuring that the resulting policy is not only successful but also energy-efficient, consistent, and suitable for physical deployment.

4.2 Robot Selection

The selection of a target platform for learning recovery behaviors was driven by a combination of physical design, simulation fidelity, and real-world feasibility. We initially considered two robots: the Unitree H1 and the Booster T1.

The H1 is a 1.7-meter-tall humanoid designed for industrial tasks, with a relatively heavy build (47 kg) and a structure optimized for cost-effective mass production. In contrast, the T1 is a smaller (1.2 meters), lighter (30 kg) robot built specifically for dynamic, athletic motion. These physical characteristics—especially its lower mass and compact frame—make the T1 more suitable for high-speed falls and repeated impact during training.

Additionally, the T1's mechanical design includes flexible joints and robust limb actuation, both of which are critical when performing full-body recovery movements. These characteristics made the T1 a more appropriate platform for our goal of developing a robust, dynamic recovery controller. The ultimate aim was to not only succeed in simulation but also pave the way for transfer to real hardware, where repeated testing and mechanical wear are significant constraints.

Table 4.1: Comparison of Booster T1 and Unitree H1 Humanoid Robots

Specification	**Booster T1**	**Unitree H1**
Height	118 cm	180 cm
Weight	Approx. 30 kg	Approx. 47 kg
Degrees of Freedom (DOF)	23 total: 6 per leg, 4 per arm, 2 head, 1 waist	18 total: 5 per leg, 4 per arm (expandable)
Joint Torque (Knee)	130 Nm	360 Nm
Walking Speed	Precise control; speed not specified	>1.5 m/s; potential >5 m/s
Battery Capacity	10.5 Ah; 2 hr walking, 4 hr standing	15 Ah (0.864 kWh); 67.2V
Sensors	Intel RealSense D455 depth camera	3D LiDAR + depth camera
Processor	14-core CPU (up to 4.8 GHz); Nvidia AGX Orin 32GB (200 TOPS)	Intel Core i5/i7; optional Jetson Orin NX
Special Features	VR capability, quick stand-up, robust build	360° depth sensing; optional dexterous hands
Price	$35,000	$120,135

4.3 Simulation Environment

To enable efficient experimentation and fast iteration, we transitioned from IsaacGym to MuJoCo Playground, an open-source learning framework that builds on the MuJoCo physics engine. Unlike IsaacGym—which is optimized for large-scale, multi-agent training on a single GPU—MuJoCo Playground is better suited for fine-tuning and evaluating single-robot policies, especially in physically demanding tasks like recovery.

One of MuJoCo Playground's most compelling features is its compatibility with multi-GPU training, which allows for scalable rollout collection and reduced wall-clock training time. The platform is also deeply integrated with JAX and supports Jupyter Notebook-based prototyping, enabling rapid debugging and on-the-fly code adjustments.

Although training speed depends heavily on hardware, we benchmarked the simulator using a simple joystick walking task across multiple GPU configurations. Results demonstrated significant speedups when using newer GPUs (e.g., RTX 3090s), reinforcing the importance of high-performance compute infrastructure.

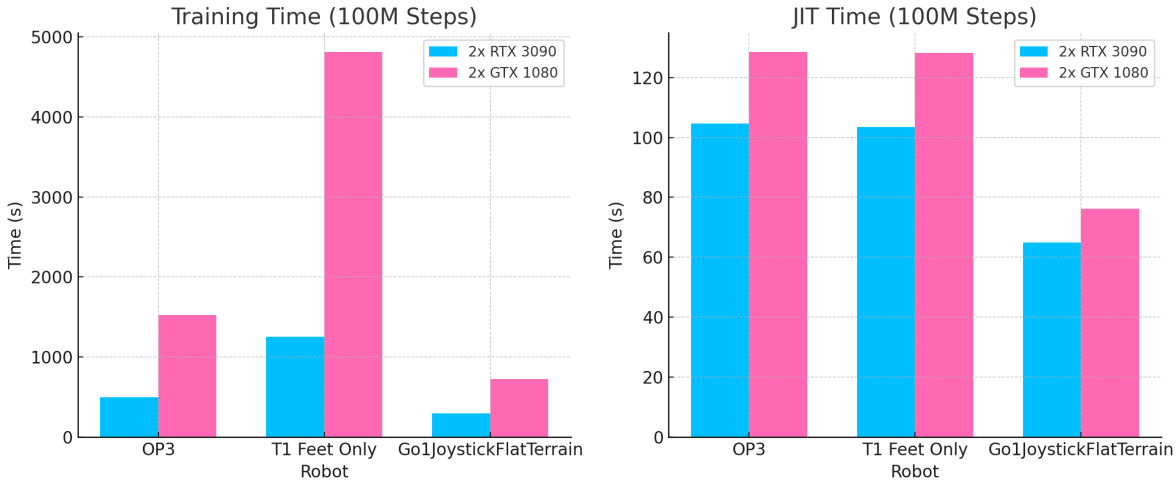

Figure 4.2: Comparison of training and JIT compilation times for three robot platforms across two GPU configurations (2× RTX 3090 and 2× GTX 1080). All experiments were run for 100 million steps.

As demonstrated in the graphs, training on the 3090s results in a 2.5–4× speedup for training time. The T1 robot is the most complex environment in the benchmark, featuring a higher-dimensional action space and more demanding contact dynamics due to its full humanoid structure. This complexity translates to longer training durations across both GPU configurations, but also highlights the benefits of accelerated hardware—training time for T1 drops from over 80 minutes on the 1080s to just over 20 minutes on the 3090s. This substantial reduction in wall-clock time not only enables faster experimentation but also makes large-scale policy iteration more feasible during development.

4.4 Environment Setup and Limitations of Baseline Recovery

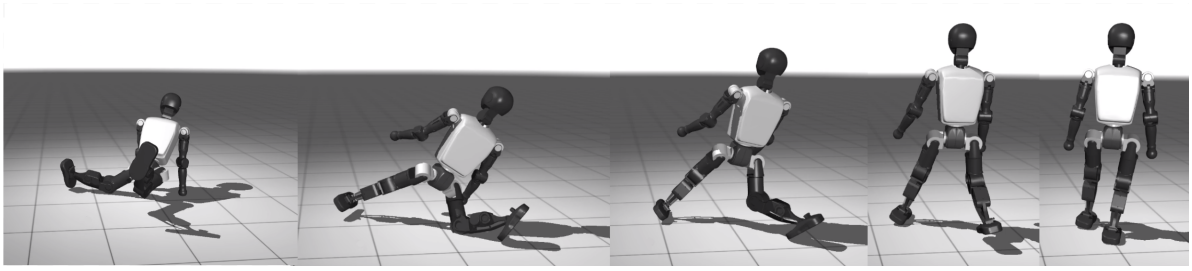

Figure 4.3: Sequence of frames from the baseline reward-based standing policy. The robot attempts to stand from a middle-split posture using primarily leg extension. While effective in limited conditions, this behavior is physically unrealistic and fragile, particularly in cluttered or uneven environments.

Initial experiments with a reward-based standing policy resulted in an overly aggressive behavior. The robot attempted to rise directly from a middle split using only its legs, particularly overloading the yaw hip joints. This strategy proved physically unrealistic and brittle; any minor degradation in motor performance caused the policy to fail. Moreover, this behavior required substantial open space and flat ground, making it unsuitable for typical indoor environments such as kitchens or warehouses.

Despite these limitations, the policy did successfully transfer to real hardware, demonstrating that even purely reward-base formulations can lead to deployable behaviors when dynamics are carefully modeled. As such, we adopt this reward-based standing controller as our baseline for comparison throughout this work.

Scripted recovery policies, while safe, lack adaptability and generalization. Our goal was to learn a robust policy capable of recovering from diverse and unpredictable fall configurations. These include realistic fall scenarios (e.g., tripping over obstacles or slipping) and adversarial ones (e.g., being pushed). RL enables learning over a distribution of fall poses, improving the robustness and adaptability of the resulting policy.

4.5 Keyframe-Based Curriculum for Safer Recovery

To avoid brittle behaviors while ensuring robust generalization, we designed a two-stage curriculum using a carefully selected intermediate keyframe. The recovery task was decomposed into two sequential stages:

- **Stage 1:** Fallen $\rightarrow$ Keyframe

- **Stage 2:** Keyframe → Standing

We tested several candidate keyframe poses inspired by natural human movements, including squatting, tabletop, downward dog, and kneeling. Each was evaluated by training separate controllers for each stage. From these poses only the squat pose was not reachable from a fallen state. The rest of the poses, although reachable from a fallen configuration, were not able to reliably transition to a standing pose except for the kneeling pose. This pose was therefore selected as the intermediate configuration for all subsequent experiments.

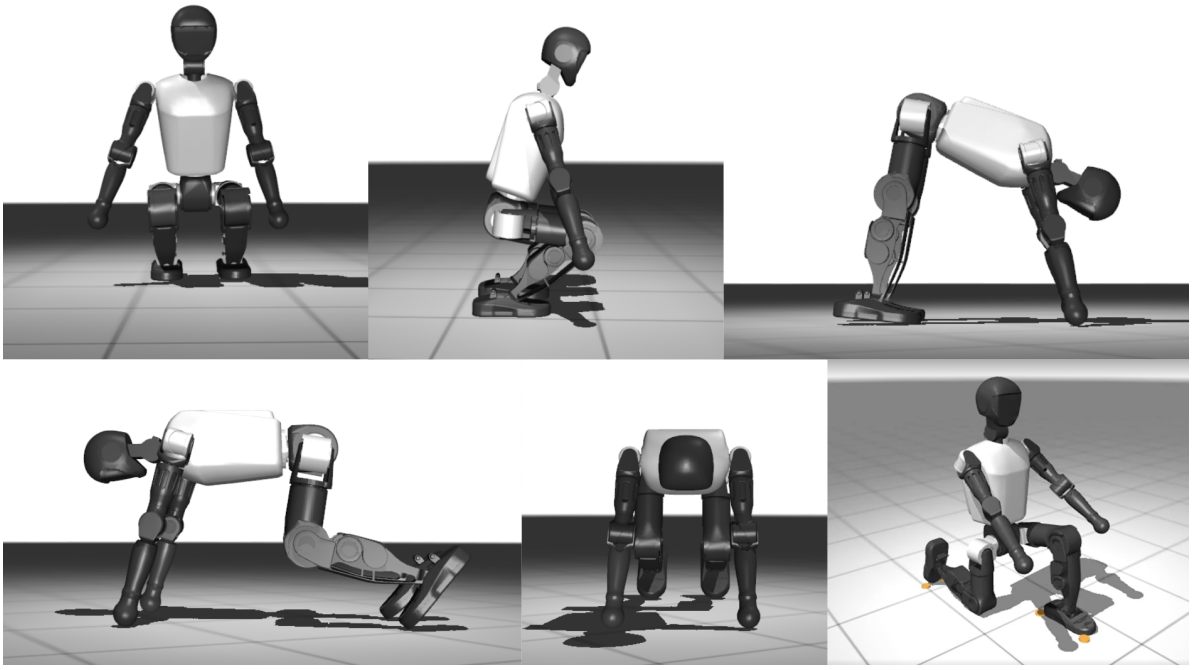

Figure 4.4: Candidate intermediate keyframe poses evaluated during curriculum design. **First row left to right:** squatting front-view, squatting side-view, downward dog **Second row left to right:** tabletop side-view, tabletop front-view, and kneeling. Each pose was tested for its ability to serve as a reliable transition point between fallen and standing configurations. Only the kneeling pose consistently enabled a successful full recovery.

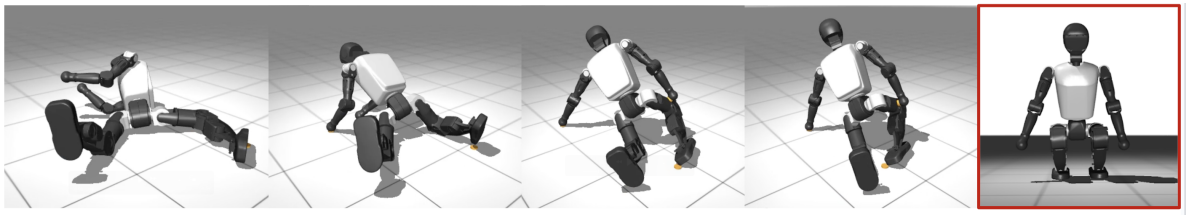

Figure 4.5: Sequence of frames from the fallen-to-squat reward-based policy. The robot is unable to achieve the final squat pose shows for reference in the red box. The robot gets stuck in a kneeling position.

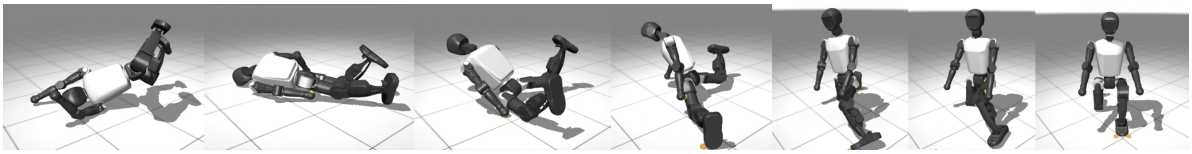

Figure 4.6: Sequence of frames from the fallen-to-kneeling reward-based policy. The robot successfully gets to the kneeling pose from a random fallen position.

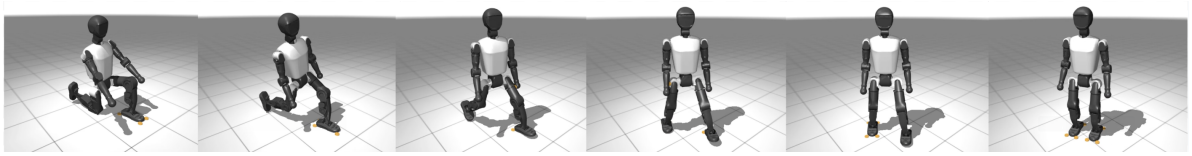

Figure 4.7: Sequence of frames from the kneeling-to-stand reward-based policy. The robot successfully gets to the standing pose from the kneeling pose.

We experimented with training the full getup controller using fine-tuning from two different initialization points: (1) a policy pre-trained on the keyframe-to-standing transition, and (2) a policy pre-trained on the fallen-to-keyframe motion. We found that only the policy fine-tuned from the keyframe-to-standing checkpoint was able to successfully learn the full getup behavior. This highlights the importance of initializing from a policy that has already mastered the final, more challenging standing motion.

The final recovery controller was then fine-tuned to execute the full recovery sequence—from fallen to kneeling to standing—by initializing from all three configurations with sampling probabilities of 0.4, 0.4, and 0.2, respectively.

4.6 Observations for Getup Controller Training

The recovery controller maintained a deliberately limited observation space throughout all stages of training. Specifically, it utilized only a small set of core physical signals: the gyroscope, gravity vector, joint angle offsets, and joint velocities. These signals were chosen to represent the minimal proprioceptive information necessary for reliable recovery behavior, without relying on external sensory input.

Observation	**Dimension**
Gyroscope	3
Gravity vector	3
Joint angle offsets	23
Joint velocities	23
Total	**52**

Table 4.2: Observation space used during training of the getup controller.

We found this observation space to be the bare minimum required for learning robust recovery behaviors. Notably, the controller does not utilize additional proprioceptive data such as end-effector positions or contact forces.

After the initial training phase, a finetuning stage was conducted to improve robustness. During this phase, uniform random noise in the range $[-1,1]$ was added to each observation entry. This noise injection strategy served to improve generalization and tolerance to sensor imperfections during real-world deployment.

4.7 Reward Design

We employed a reward function composed of three components: a height and orientation reward, a pose reward, and an energy cost penalty.

Height & Orientation Reward

This reward encourages the robot to maintain an upright posture and appropriate vertical stance. It penalizes deviations below the desired torso and waist height, and rewards alignment of the torso with the gravitational up vector.

Reward Function Variables

$$\begin{aligned} h_{\text{torso}} &: \text{ actual torso height} \\ h_{\text{waist}} &: \text{ actual waist height} \\ h^{\text{des}}_{\text{torso}},\ h^{\text{des}}_{\text{waist}} &: \text{ desired torso and waist heights} \\ u_z &: \text{ z-component of the torso's up vector} \end{aligned}$$

Reward Function Formula

$$\epsilon_t = \text{clip}\left(\frac{h^{\text{des}}_{\text{torso}} - \min(h_{\text{torso}}, h^{\text{des}}_{\text{torso}})}{h^{\text{des}}_{\text{torso}}},\ 0,\ 1\right), \quad r_{\text{torso}} = 1 - \epsilon_t$$

$$\epsilon_w = \text{clip}\left(\frac{h^{\text{des}}_{\text{waist}} - \min(h_{\text{waist}}, h^{\text{des}}_{\text{waist}})}{h^{\text{des}}_{\text{waist}}},\ 0,\ 1\right), \quad r_{\text{waist}} = 1 - \epsilon_w$$

This form ensures that If the actual height $\mathbf{h}$ exceeds the desired height $\mathbf{h}_{\text{des}}$, the error is zero and if the height falls short, the error scales proportionally to how far it is from the target. The clipping between 0 and 1 prevents overly harsh penalties and ensures numerical stability.

The robot's uprightness is measured using the z-component of its torso's up vestor, $\mathbf{u}_{\text{z}}$, which ranges from -1 (completely upside down) to +1 (perfectly upright).

$$r_{\text{ori}} = (0.25u_z + 0.5)^2$$

Squaring the result adds curvature, sharply penalizing intermediate tilts while strongly rewarding upright alignment.

$$r_{\text{height}} = r_{\text{torso}} + r_{\text{waist}}$$
$$\text{reward} = \frac{2 \cdot r_{\text{ori}} \cdot (r_{\text{height}} + 1)}{3}$$

Pose Matching Reward

The pose reward encourages the robot to match a reference pose (either kneeling or standing) by minimizing the squared error between the current and desired joint configurations.

Let $\mathbf{q}_{\text{curr}}$ be the current joint configuration and $\mathbf{q}_{\text{ref}}$ be the reference configuration. Then:

$$\text{error} = \|\mathbf{q}_{\text{curr}} - \mathbf{q}_{\text{ref}}\|_2^2,$$
$$\text{reward} = \exp\left(-0.5 \cdot \text{error}\right)$$

This results in a smooth, exponentially decaying reward that is maximal when the pose is a perfect match.

Energy Efficiency Penalty

To encourage smooth, efficient behavior, we penalize the total absolute mechanical power output by the actuators. Given joint velocities v_i and normalized actuator torques $\tau_i/\tau_i^{\max}$, the energy cost is defined as:

$$c_{\text{energy}} = \sum_{i=1}^{n} \left| v_i \cdot \frac{\tau_i}{\tau_i^{\max}} \right|$$

We initially trained the policy using only the height and pose rewards, and later fine-tuned the controller with the energy cost penalty to tame high-effort behaviors.

4.8 Gating Conditions for Transitioning to Standing

To train the full getup controller, we introduce a gating mechanism that determines when to transition from the keyframe pose to the standing phase. This is implemented by adding a gate-dependent term to the reward function. When the gate condition is met, the robot receives the reward associated with transitioning from the keyframe pose to the standing pose, plus a small constant bonus. When the gate condition is not met, the robot instead receives the reward from the fallen-to-keyframe controller.

$$\begin{aligned}\text{Height Reward} = &\underbrace{(1 - \texttt{gate}) \cdot \texttt{reward\_height}(h_{\text{torso}}, h_{\text{waist}}, \vec{g})}_{\text{Keyframe reward}} \\ &+ \underbrace{\texttt{gate} \cdot (1.83 + \texttt{r\_height}(h_{\text{torso}}, h_{\text{waist}}, \vec{g}, 0.67, 0.55))}_{\text{Standup reward}},\end{aligned}$$

$$\begin{aligned}\text{Pose Reward} = &\underbrace{(1 - \texttt{gate}) \cdot \texttt{reward\_pose}(\vec{q})}_{\text{Keyframe reward}} \\ &+ \underbrace{\texttt{gate} \cdot (0.6 + 100 \cdot \texttt{r\_pose}(\vec{q}, \vec{q}_{\text{init}}))}_{\text{Standup reward}}.\end{aligned}$$

The small constant bonus plays an important role in encouraging the robot to progress through the entire getup pipeline. Without it, the reward obtained from remaining in the fallen-to-keyframe phase might exceed the initial reward from the keyframe-to-standing phase, disincentivizing the policy from triggering the transition. This adjustment prevents the robot from stalling just before initiating the final stand-up motion.

The gate condition itself consists of two components: a height check and an uprightness check.

The height gate checks whether the robot's torso has reached (or exceeded) a predefined target height—in this case, 0.4 meters. To ensure the gate condition remains stable as the

robot continues to rise, the function computes the minimum of the current torso height and the target height. The gate then calculates the error as the difference between the target height and this clamped value. If the error is less than 0.005 meters (i.e., within 5 mm), the height condition is satisfied.

The upright gate assesses whether the robot's orientation aligns with the global vertical axis. Specifically, it computes the Euclidean norm (L2 distance) between the robot's up vector—typically derived from an onboard IMU—and the global up direction [0, 0, 1]. If this deviation is less than 0.01, the robot is considered sufficiently upright.

4.9 Comparison to Baseline Getup

To evaluate the performance and efficiency of the proposed keyframe-based getup controller, we compared it against a baseline learned getup policy across a variety of metrics. Each policy was deployed from the same fallen initial condition across 1000 random seeds (both start from the same initial fallen state, each fallen state is different for each different seed) to ensure statistical significance and robustness of the results.

The metrics capture energy efficiency (energy sum and max energy), physical stress on the robot (torques, joint accelerations), and stability indicators (contact forces). These metrics were recorded per rollout and then averaged across all seeds for comparison.

Table 4.3: Comparison between baseline and keyframe-based getup policies (averaged over 1000 seeds).

Metric	**Baseline**	**Keyframe**
Average Energy Sum	2279.839	**1604.966**
Average Max Torques	438.703	**420.283**
Average Max Joint Acceleration	425.661	**301.457**
Total Joint Acceleration	1797.189	**1313.935**
Total Torques	37386.540	**39763.719**
Average Contact Force	**449.743**	857.012
Z Contact Force (Total)	4768.910	**4714.383**
Average Max Energy	96.558	**80.327**

The keyframe-based getup controller demonstrates consistent improvements over the baseline across nearly all evaluation metrics, indicating a more efficient and mechanically favorable recovery behavior. Most notably, it requires substantially less energy to complete the getup motion, achieving a 29.6% reduction in the average energy sum compared to the baseline. This suggests a more deliberate and optimized trajectory through the keyframe motion.

In addition to improved energy efficiency, the keyframe controller exhibits significantly reduced peak and cumulative joint accelerations. These lower values imply smoother, more

controlled limb trajectories that are less likely to induce mechanical wear or instability. The average maximum energy exerted during the motion is also lower, indicating that the keyframe controller avoids the kind of energy spikes that can signal unstable or abrupt control behaviors.

Although the total torque usage is slightly higher for the keyframe controller, the maximum torque observed is lower, suggesting a more even distribution of actuation effort throughout the motion. This tradeoff favors durability, as lower peak torques reduce stress on the actuators. One area where the keyframe controller appears to underperform is in average contact force, which is nearly doubled compared to the baseline. However, the total vertical (z-axis) contact force remains slightly lower, indicating that the robot maintains vertical stability while possibly relying more heavily on lateral or distributed contact forces. This can be interpreted as the robot leveraging the ground more actively for support during the transition, which may contribute to the smoother and more energy-efficient motion observed.

Overall, the keyframe-based getup strategy offers a robust and physically conservative alternative to learned policies, making it particularly well-suited for deployment on physical hardware where safety, energy efficiency, and mechanical longevity are critical.

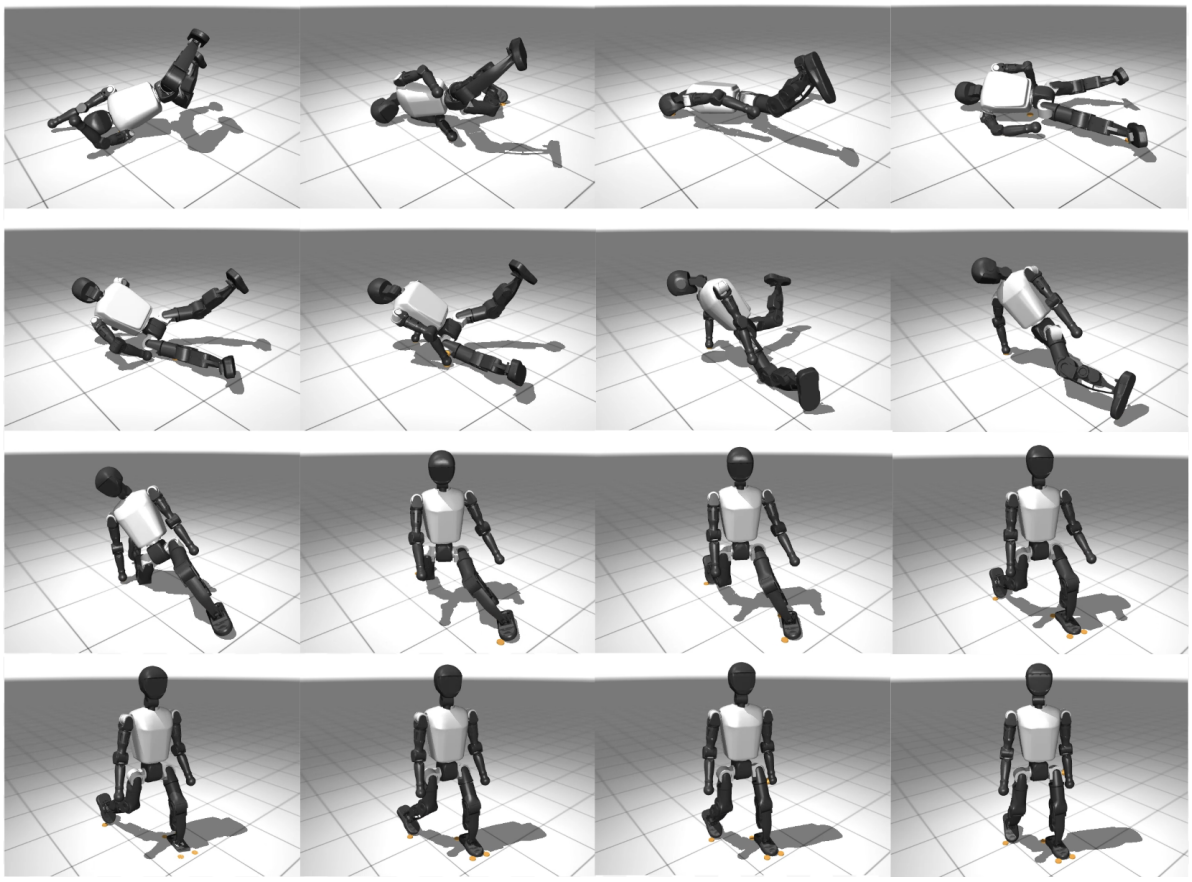

Figure 4.8: Keyframe policy full getup sequence.

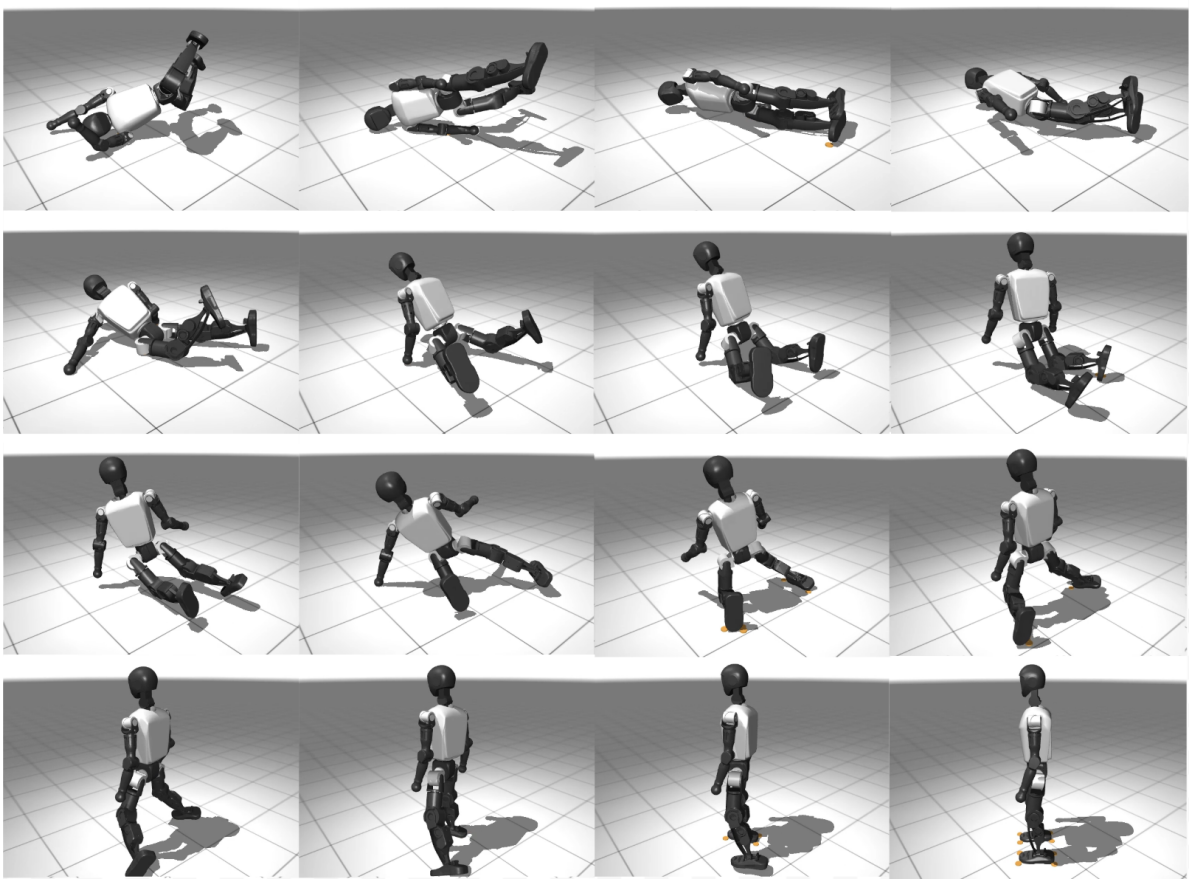

Figure 4.9: Baseline policy full getup sequence.

Chapter 5

Skill Integration for End-to-End Humanoid Control

Motivation for Policy Integration

Having trained a robust getup controller capable of consistent and energy-efficient recovery, our next objective is to develop a *unified policy* that integrates both walking and recovery into a single, autonomous system. This is motivated by the ultimate goal of deploying humanoid robots in real-world environments, where they must not only locomote efficiently but also recover from inevitable failures—without requiring manual intervention, reset mechanisms, or brittle state machines.

A fully deployable controller must determine, in real time, *when to walk and when to recover*, based solely on its onboard observations. This is especially important in dynamic or cluttered settings, where external triggers or scripted resets are impractical or unsafe. The robot must learn to reason about its own physical state and transition fluidly between behaviors. In short, we seek to build a *single end-to-end controller* that can walk, fall, recover safely, and continue walking—all without hard-coded transitions or external supervision.

This chapter describes our approach to fusing locomotion and recovery skills into a single policy using a teacher-student learning framework, bolstered by DAgger for robust online imitation learning. Our integrated controller emphasizes the same principles that guided our recovery design: safety, consistency, and physical plausibility. Rather than relying on abrupt control switches or redundant policy branches, the goal is to produce a unified network that reasons holistically about body posture and task objectives. This integration ensures that transitions between standing and fallen states are smooth, humanlike, and resilient to perturbations.

We begin by describing the locomotion environment and challenges of direct multitask learning, then introduce our solution using supervised distillation and dataset aggregation. Our design choices reflect the core motivation of this thesis: building controllers that are not only effective in simulation but suitable for deployment on real-world humanoid hardware, where reliability and safety are paramount.

Locomotion Policy Setup

To construct the walking component, we utilize the same joystick-based locomotion environment for the T1 robot as presented in the original MuJoCo Playground paper. Following the published specifications and publicly available codebase, we retrain a joystick policy capable of tracking a velocity command comprising forward velocity, lateral velocity, and desired yaw rate.

The resulting locomotion policy demonstrates robust performance, reliably achieving accurate velocity tracking across a variety of terrains and environmental conditions. With strong locomotion and getup behaviors separately learned, the next challenge is how to *combine them into a single policy.*

Challenges of Direct Multi-Behavior Learning

Training a monolithic policy that learns both getup and locomotion behaviors simultaneously is *highly nontrivial.* Such multitask training can lead to:

- Unstable convergence,
- Slow learning dynamics,
- Reward engineering difficulties, and
- Poor transferability between skill domains.

Directly learning both behaviors jointly risks mode collapse toward the easier behavior (e.g., walking) or catastrophic forgetting of one of the skills during training.

Teacher-Student Framework

To address this, we adopt a *teacher-student training framework* that divides the learning process into two distinct phases:

- **Phase 1: Construct a Teacher Policy**
  We first create a robust teacher policy by naively stitching together the getup and locomotion controllers. At every timestep, the teacher policy queries both the getup and locomotion policies and selects the action to execute based on a simple condition evaluated on the current state. In our case, the *waist height* of the robot serves as a reliable indicator: if the waist is below a threshold, the robot is considered fallen and the getup controller is used; otherwise, the walking controller is selected.
- **Phase 2: Train a Student Policy via Supervision**
  Rather than train directly on pre-collected trajectories, the student policy is supervised to imitate the teacher's behavior through continuous online learning, thereby avoiding exposure bias.

Dataset Aggregation (DAgger)

Rather than employing a *naive imitation learning* approach—which would involve training the student only on static, pre-recorded demonstrations—we utilize *Dataset Aggregation (DAgger)*. This choice is motivated by the critical shortcomings of static datasets, namely *distribution shift*: even small errors by the student can compound over time, leading it into states not represented in the teacher's data, from which recovery is unlikely.

DAgger offers an elegant solution to this problem. It is an *online learning algorithm* that actively updates the dataset during training:

- The student policy is allowed to *roll out its own trajectories*.
- At every encountered state, DAgger queries the teacher for the *correct action label*.
- The collected student-generated states paired with teacher actions are aggregated into the dataset.
- The student is then *supervised on this growing dataset*, learning not only ideal behaviors but also how to recover from its own mistakes.

Although DAgger introduces additional training time due to the need for repeated data collection, its benefits in terms of robustness and generalization significantly outweigh this cost.

Setting Up DAgger in MuJoCo Playground

Implementing DAgger within MuJoCo Playground required careful consideration of observation space compatibility between the getup and locomotion teacher policies. Each teacher policy had been trained with different observation inputs, leading to challenges in constructing a unified environment that could serve both policies while training the student.

Specifically, the getup policy and locomotion policy operated on partially overlapping but distinct observation spaces:

As shown in Table 5.1, the locomotion policy required additional observations related to task commands and foot-ground contact timing, making its input space a strict superset of the getup policy's inputs.

To accommodate both policies during training:

- The **teacher policy** was implemented to collect the union of observations, allowing it to correctly query either the getup or locomotion subpolicy based on the appropriate input subset.
- The **student policy** was trained on the full, combined observation space—matching the union of both teacher inputs—to ensure it received all necessary sensory information to imitate either behavior.

Table 5.1: Comparison of Getup and Locomotion Policy Observation Spaces

Observation Feature	**Getup Policy**	**Locomotion Policy**
Gyroscope (angular velocity)	✓	✓
Gravity vector	✓	✓
Joint angle offsets	✓	✓
Joint velocities	✓	✓
Last action	✓	✓
Linear velocity	–	✓
Locomotion command (desired velocity)	–	✓
Foot phase indicator	–	✓

This setup allowed seamless switching between the getup and locomotion policies during training rollouts, and enabled the student to learn a single unified mapping from rich observations to actions, covering both locomotion and recovery behaviors.

Algorithm 1 DAgger (Dataset Aggregation)

Require: Environment `env`, Teacher policy π_{teacher}, `hyperparameters`

Initialize π_{student}
Initialize dataset $\mathcal{D} \leftarrow \emptyset$
// Main Training Loop
for iteration $i = 1$ **to** `dagger_iterations` **do**
    $\beta \leftarrow$ `lambda step: jp.where(step == 0, 1.0, 0.0)` ▷ probability of following teacher
    // Data Collection Phase
    for `step` in range `environment steps` **do**
        Sample action a_t:

$$a_t = \begin{cases} \pi_{\text{teacher}}(s_t) & \text{with probability } \beta \\ \pi_{\text{student}}(s_t) & \text{else} \end{cases}$$

        Execute a_t in environment to transition to s_{t+1}
        Record $(s_t, \pi_{\text{teacher}}(s_t))$ into $\mathcal{D}$ ▷ s_t labeled with teacher's action
    end for
    // Supervised Training Phase
    for `epoch` in range `epochs` **do**
        for each mini-batch from $\mathcal{D}$ **do**
            Update π_{student} to minimize:

$$\mathcal{L} = \|\pi_{\text{student}}(s) - \pi_{\text{teacher}}(s)\|^2$$

        end for
    end for
end for
return π_{student}, metrics

Student Policy Training

To ensure stable and efficient DAgger training, we selected hyperparameters tailored to the dual-skill distillation task. A large number of parallel environments (1024) was used to accelerate data collection across diverse robot states, while the episode length (1000 steps) and demonstration length (6 steps) ensured that fall recovery events could be accurately sampled. The DAgger β-schedule was defined as a step function, starting with $\beta = 1.0$ to exclusively query the teacher during initial exploration and transitioning immediately to $\beta = 0.0$ to encourage the student to act independently in subsequent iterations. This setup focuses student learning on correcting its own mistakes without early-stage overreliance on the teacher. We used a batch size of 256 and a learning rate of 4×10^{-4} for stable supervised learning updates. The number of DAgger steps (100,000) was carefully aligned with the evaluation interval to ensure regular monitoring without biasing the dataset toward only early training stages. Overall, these hyperparameters were selected to balance fast convergence with robustness against distribution shift during policy distillation.

Table 5.2: Hyperparameters used for DAgger training.

Parameter	**Value**	**Notes**
Domain randomization	False	Optional
Number of environments	1024	Parallel data collection
Episode length	1000	Steps per episode
DAgger steps	100k	Adjusted to evaluation schedule
Number of evaluations	10	Per DAgger iteration
Demonstration length	6	Short expert episodes
Epochs per phase	4	Per supervised training phase
DAgger β schedule	$1.0 \rightarrow 0.0$	Step function
Batch size	256	For student updates
Learning rate	4×10^{-4}	Adam optimizer
Normalize observations	True	Across dataset
Tanh squash actions	False	Raw actions output
Scramble time	1000	Observation shuffling
Evaluation rollout length	1150	1.15×episode length

To further evaluate the effectiveness of the training schedule, we tested the behavior of the distilled policy across different numbers of DAgger steps, comparing models trained with 100,000, 1,000, and 400 DAgger steps to the baseline teacher policy performance. This allowed us to analyze how the amount of imitation supervision affects the final policy quality relative to the expert.

The following figures compare the performance of the student and teacher policies in terms of torso height, waist height, and forward linear velocity. Each policy was deployed from the same fallen configuration using the same seed (62) and commanded to follow a

constant velocity input of [0.4, 0, 0]. These comparisons illustrate how each policy responds when tasked with recovering and walking forward under identical initial conditions.

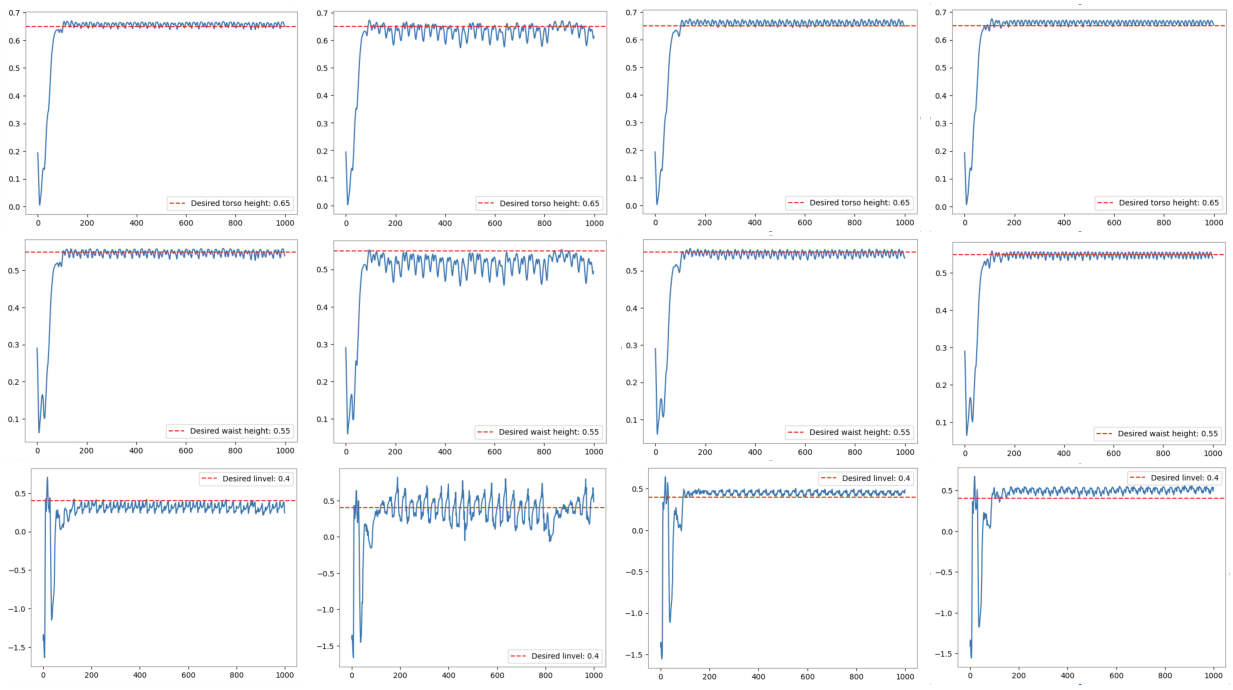

Figure 5.1: Comparison of torso height (top), waist height (middle), and forward velocity (bottom) across different policies (left to right): Teacher, DAgger-400, DAgger-1000, and DAgger-100k. All policies were evaluated from a fallen state with a velocity command of [0.4, 0, 0].

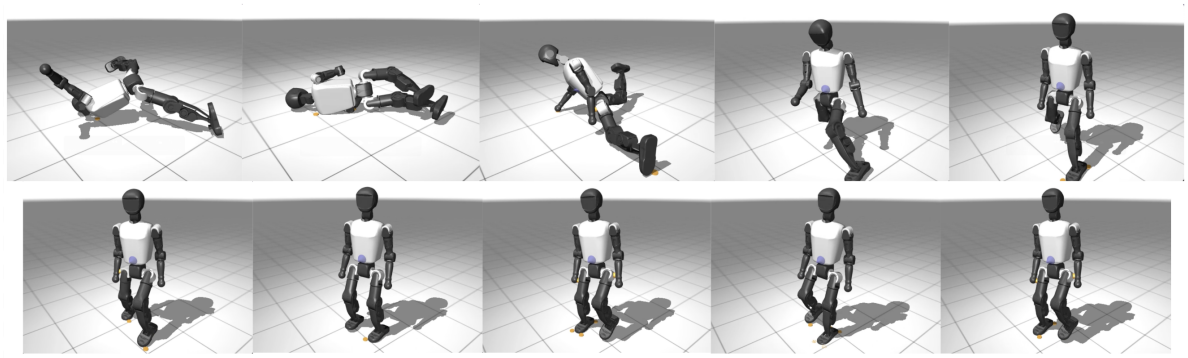

Figure 5.2: Sequence of frames from the teacher policy. The robot successfully starts walking after getting up from the fallen position. The blue arrow coming out of the torso is a visualization of the command direction and magnitude, here 0.4 m/s forward. The robot walks with an even gait.

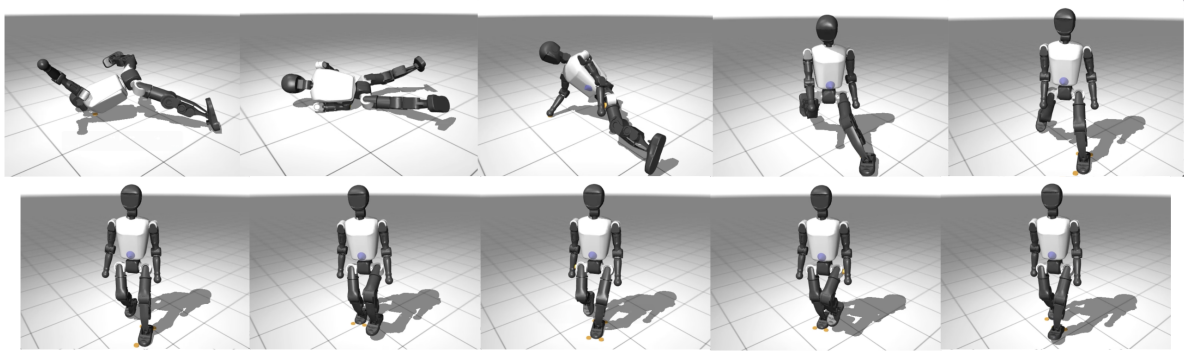

Figure 5.3: Sequence of frames from the DAgger-400 policy. The robot successfully starts walking after getting up from the fallen position. The blue arrow coming out of the torso is a visualization of the command direction and magnitude, here 0.4 m/s forward. The robot steps forward by stepping the left foot out and then stepping the right foot closed. The right foot never passes the left foot leading to an uneven gait

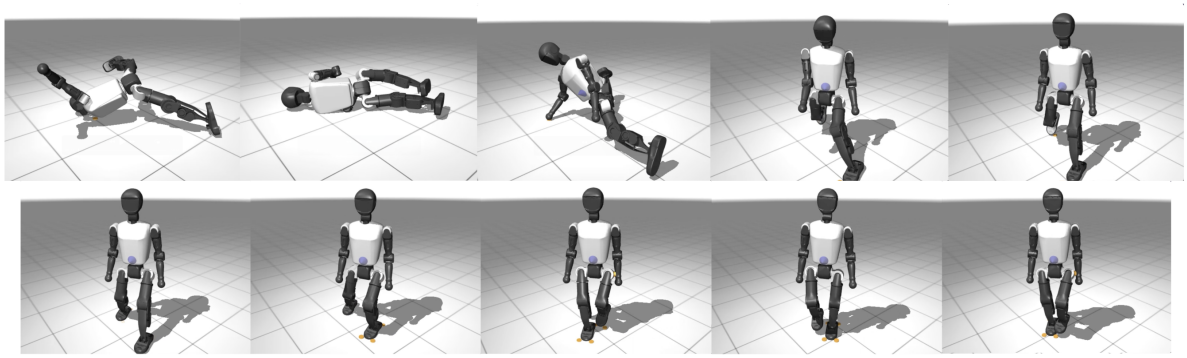

Figure 5.4: Sequence of frames from the DAgger-1000 policy. The robot successfully starts walking after getting up from the fallen position. The blue arrow coming out of the torso is a visualization of the command direction and magnitude, here 0.4 m/s forward. The robot walks with an even gait.

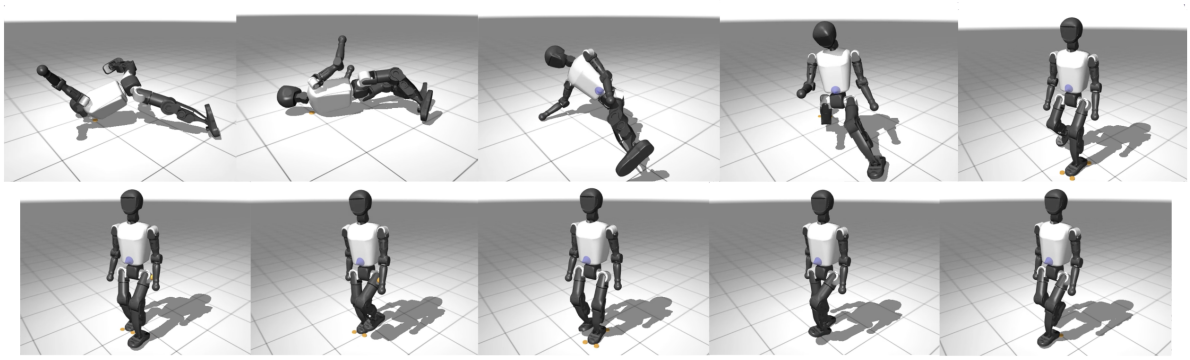

Figure 5.5: Sequence of frames from the DAgger-100k policy. The robot successfully starts walking after getting up from the fallen position. The blue arrow coming out of the torso is a visualization of the command direction and magnitude, here 0.4 m/s forward. The robot walks with an even gait.

As shown in the figures above, the DAgger-1000 and DAgger-100k policies perform nearly indistinguishably from each other and closely match the performance of the teacher policy. In contrast, the DAgger-400 policy exhibits poorer performance, characterized by an asymmetric and less stable gait. However, when deployed in the MuJoCo sim-to-sim environment, the DAgger-100k policy demonstrates noticeably more stable behavior compared to DAgger-1000. Based on this observation, we elect to proceed with the DAgger-100k policy, despite the DAgger-1000 appearing comparably effective in earlier evaluations. This decision comes at the cost of significantly increased training time: the DAgger-100k policy required 8 hours,

28 minutes, and 45 seconds to train, whereas DAgger-1000 training completed in only 14 minutes and 45 seconds.

Chapter 6

Evaluation

6.1 Evaluation Set-Up

To rigorously assess the robustness, consistency, and real-world suitability of the unified DAgger-100k policy, we evaluate it on three environments that were *unseen during training*. Each evaluation environment was carefully designed to mimic challenges a humanoid robot might realistically encounter, including falls, slipping, or tripping over objects. These evaluations are not just about whether a robot can eventually succeed, but how reliably, safely, and efficiently it performs—hallmarks of a controller that could be deployed in the real world.

Baseline Policy

To evaluate performance against a baseline, we construct a composite policy that combines the baseline getup controller with the joystick-based locomotion policy used during DAgger training. This stitching is performed using the same switching mechanism employed for the teacher policy during the DAgger training. Specifically, control is delegated to the getup policy whenever the robot's waist height falls below 0.5 meters, and to the locomotion policy otherwise. We refer to this combined controller as the *baseline policy* throughout our evaluation.

Selection of Evaluation Environments

The three environments chosen for evaluation are:

- **Low-Wall Environment**
- **Puddle Environment**
- **Push Environment**

These scenarios were selected to stress different aspects of the policy's design, such as obstacle negotiation, slip recovery, and resilience to strong disturbances. Importantly, they also test the benefits of the keyframe-based recovery strategy in scenarios where prior scripted or reward-shaped behaviors often fail or behave erratically.

Low-Wall Environment

The **low-wall environment** challenges the robot's ability to detect and react to small but significant obstacles. It consists of a series of three low walls, each spaced 6 meters apart in front of the robot. Each wall is 0.2 meters tall, 0.1 meters thick, and extends 40 meters in length. The exaggerated wall length ensures that the robot cannot simply walk around the obstacle and must confront it directly.

Importantly, the height of the walls is just below the robot's knee joint, making them difficult to step over and easy to trip on if not properly accounted for. This setup is intended to simulate realistic environmental hazards, such as an unnoticed box or low barrier on the ground, where a robot might attempt to walk through the object rather than around it, leading to a fall.

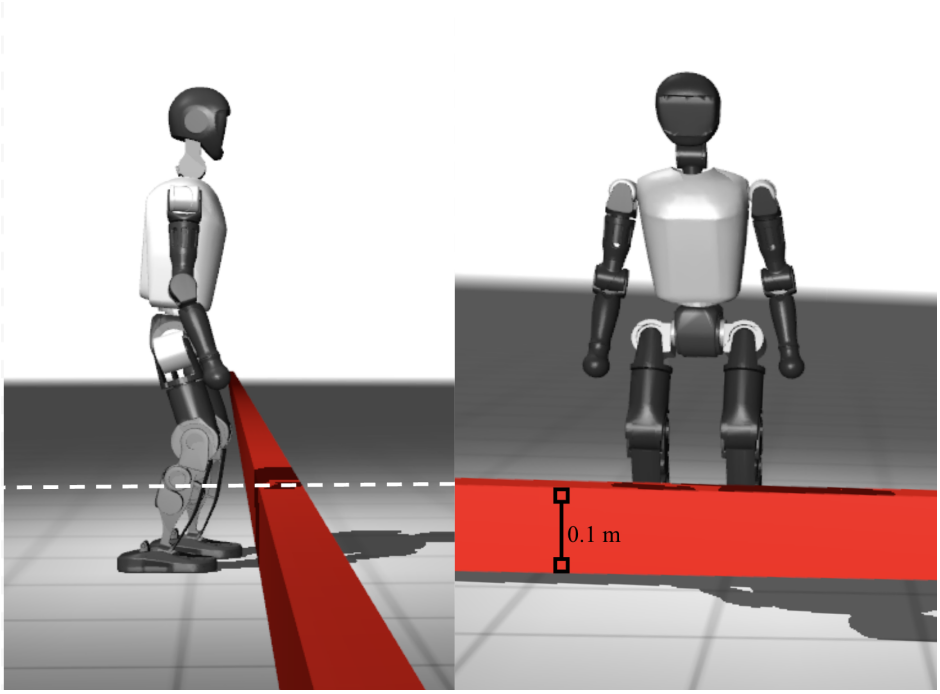

Figure 6.1: The T1 robot standing next to a low wall for height comparison. The left image shows a side view, with a dashed white line indicating where the wall meets the robot's lower leg. The right image provides a front view, with a line marking the wall's height of 0.1 meters.

To compare the performance of the DAgger-100k policy against the baseline policy, we designed a timed evaluation in this challenging environment containing three consecutive low walls. The robot begins 2 meters from the first wall, with each subsequent wall spaced 6 meters apart, resulting in a total distance of 14 meters. In each trial, the robot is required to walk forward, encounter and recover from each wall, and ultimately reach the final location, at which point it must successfully get up and pace in place.

Each policy was tested across five trials. Completion time was measured from the start of the episode until the robot successfully stood up after the third wall. All evaluation rollouts are provided in Appendix A.The DAgger-100k policy consistently completed all five trials, achieving a 100% success rate. In contrast, the baseline policy failed to complete one of the five trials, resulting in an 80% success rate. While the baseline policy demonstrated a slightly faster average completion time (35.25 seconds vs. 37.8 seconds), its reduced reliability is a critical limitation—particularly in real-world deployments where consistent recovery is essential.

Table 6.1: Trial Results for DAgger-100k Policy

Trial	**Completion Time (s)**	**Getup Attempts (Total)**	**Avg. Attempts per Wall**
1	38	3	1.00
2	38	5	1.67
3	38	4	1.33
4	37	4	1.33
5	38	3	1.00
Average	**37.8**	–	**1.26**

Table 6.2: Trial Results for Baseline Policy

Trial	**Completion Time (s)**	**Getup Attempts (Total)**	**Avg. Attempts per Wall**
1	37	4	1.33
2	33	5	1.67
3	*DNF*	12	–
4	34	7	2.33
5	37	7	2.33
Average	**35.25**	–	**2.5**

* In Trial 3, the robot did not finish (DNF) the course. The 12 getup attempts refer to the first two walls, which it successfully recovered from.

While the baseline policy demonstrated slightly faster traversal in successful attempts, it required significantly more getup attempts on average (2.5 vs. 1.26) and failed to recover in one trial. Given the demands of real-world deployment, robustness and reliability take precedence over marginal speed gains. Thus, the DAgger-100k policy's perfect success rate and more efficient recovery behavior mark a clear advantage in practical settings.

Puddle Environment

The **puddle environment** introduces randomized patches of low-friction terrain to evaluate the robot's ability to maintain balance and recover from slips—an essential capability for real-world navigation where surface properties can vary unpredictably. These "puddles" are modeled as thin geometric surfaces with a friction coefficient of 0.4 and vary in size from 0.6 to 1.0 meters in both width and length. This irregular configuration simulates common hazards such as wet floors, mud, or spilled liquids in unstructured environments.

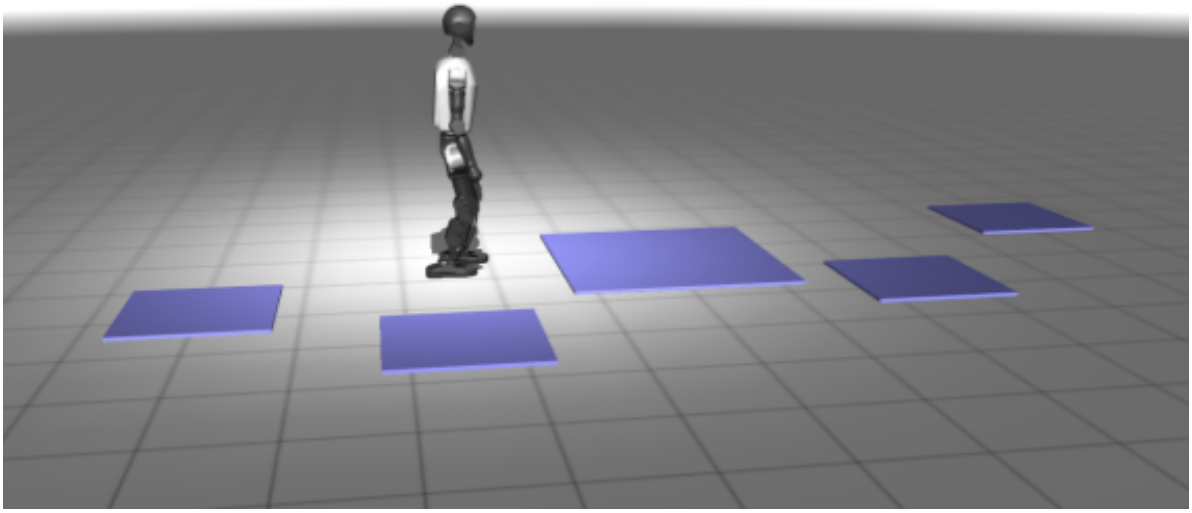

Figure 6.2: The T1 robot standing in the puddle environment.

To evaluate recovery behavior in this setting, we conducted 10 trials for each policy variant (Baseline and DAgger-100k), issuing a constant forward velocity command in all cases. Performance was assessed based on whether the robot slipped, recovered, and how many getup attempts were required. All evaluation rollouts are provided in Appendix B.

DAgger-100k Results:

The DAgger-100k policy experienced slips in 3 of the 10 trials (Trials 2, 3, and 8). Recovery was successful in all three cases, requiring 1, 1, and 3 getup attempts respectively. Additionally, the policy exhibited several preemptive stabilization behaviors not observed in the baseline: in one trial, the robot recovered from an incipient slip by taking a backward step, and in two other trials, it re-centered its feet to regain balance—without deviating from the forward command. These behaviors demonstrate a level of adaptive, human-like reflex not explicitly programmed into the controller.

Table 6.3: DAgger-100k Performance in Puddle Environment

Trial	Slipped?	Getup Attempts
2	Yes	1
3	Yes	1
8	Yes	3
Others	No	–

Baseline Policy Results:

The baseline policy slipped in 2 of the 10 trials (Trials 6 and 10), requiring 3 and 2 getup attempts respectively. Unlike the DAgger policy, no anticipatory or corrective behaviors were observed prior to the slips. The baseline controller consistently executed its locomotion pattern without modulating stride or contact timing in response to terrain feedback.

Table 6.4: Baseline Performance in Puddle Environment

Trial	Slipped?	Getup Attempts
6	Yes	3
10	Yes	2
Others	No	–

Discussion: Slipping Frequency vs. Recovery Robustness

While the DAgger-100k policy exhibited a higher slip rate (3 slips vs. 2 in the baseline), its recovery behavior was more consistent, efficient, and physically grounded. The DAgger controller not only completed more of its getups in fewer attempts on average, but also demonstrated proactive balance correction strategies—such as stepping back or adjusting foot placement—without any change in the commanded velocity. These emergent behaviors suggest a more nuanced internal representation of balance and terrain response.

In contrast, the baseline's lower slip count may reflect a stronger underlying locomotion policy, potentially due to performance degradation introduced during DAgger integration. However, the baseline's inability to adapt preemptively or recover gracefully underscores a

critical limitation for real-world deployment. The reliance on torque-heavy or repeated attempts increases energy consumption, hardware strain, and environmental unpredictability.

Thus, despite a marginal increase in slip events, the DAgger-100k policy remains preferable due to its superior resilience, adaptive recovery, and safety-critical behavior patterns under low-friction conditions.

Push Environment

The **push environment** applies force impulses directly to the robot's torso, simulating both intentional and accidental collisions. This is the most direct evaluation of recovery consistency and resilience.

We conducted 20 forward push trials for each of the two policies. In every trial, the robot experienced a forward fall and was tasked with recovering to a stable upright stance with no velocity command given.

Both policies were able to eventually recover in all 20 trials. However, while the baseline required multiple, erratic attempts—averaging 3.65 attempts per fall—the DAgger-100k policy consistently recovered in nearly one attempt on average. Its worst-case performance was just 2 attempts, compared to 11 for the baseline. More critically, the baseline's recovery attempts were often uncontrolled, involving tumbling or using large amounts of space to generate forward momentum. These behaviors are unsafe for real-world environments. In contrast, the DAgger-100k controller executed smoother, more energy-efficient, and space-conserving getups—crucial traits for robots operating around people or objects.

These results highlight the improved stability and motion efficiency of the DAgger-100k policy. The most illustrative examples of failure cases for both policies are shown in Figures 6.3, 6.4, 6.5, with all trial rollouts provided in Appendix C.

Table 6.5: Summary of push recovery performance over 20 trials.

Metric	**Baseline**	**DAgger-100k**
Average Attempts per Trial	3.65	**1.16**
Maximum Attempts in Any Trial	11	**2**
Minimum Attempts in Any Trial	1	**1**
Standard Deviation	2.37	**0.37**

These results reaffirm the core thesis of this work: keyframe-based controllers inspired by human getup behavior produce safer, more stable, and more reliable recovery motions—not only succeeding more often but doing so in ways that are physically reasonable and deployment-ready.

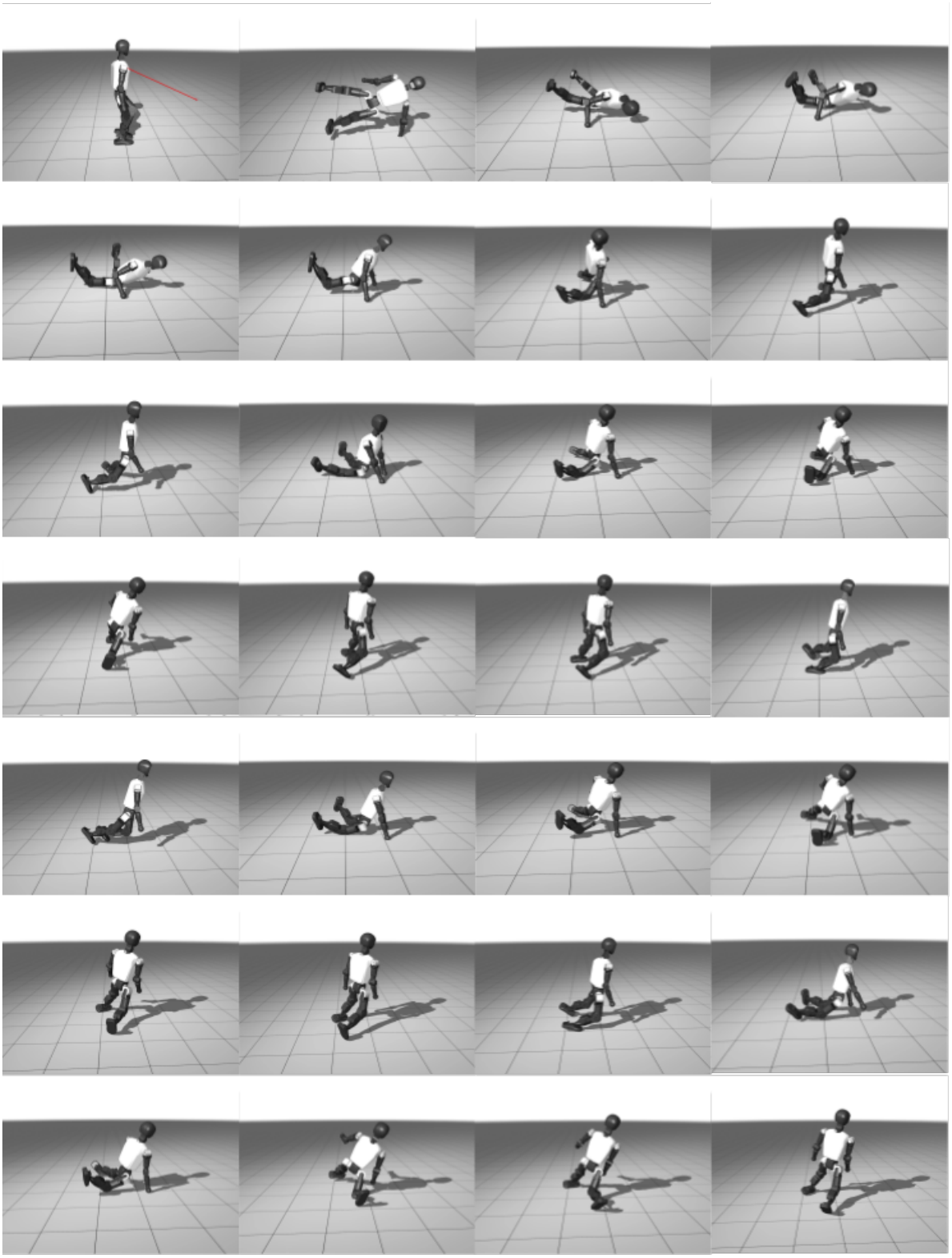

Figure 6.3: (a) The T1 robot is pushed forward from the torso, the push is visualized with the red arrow. The baseline policy attempts to getup, here are attempts 1-4.

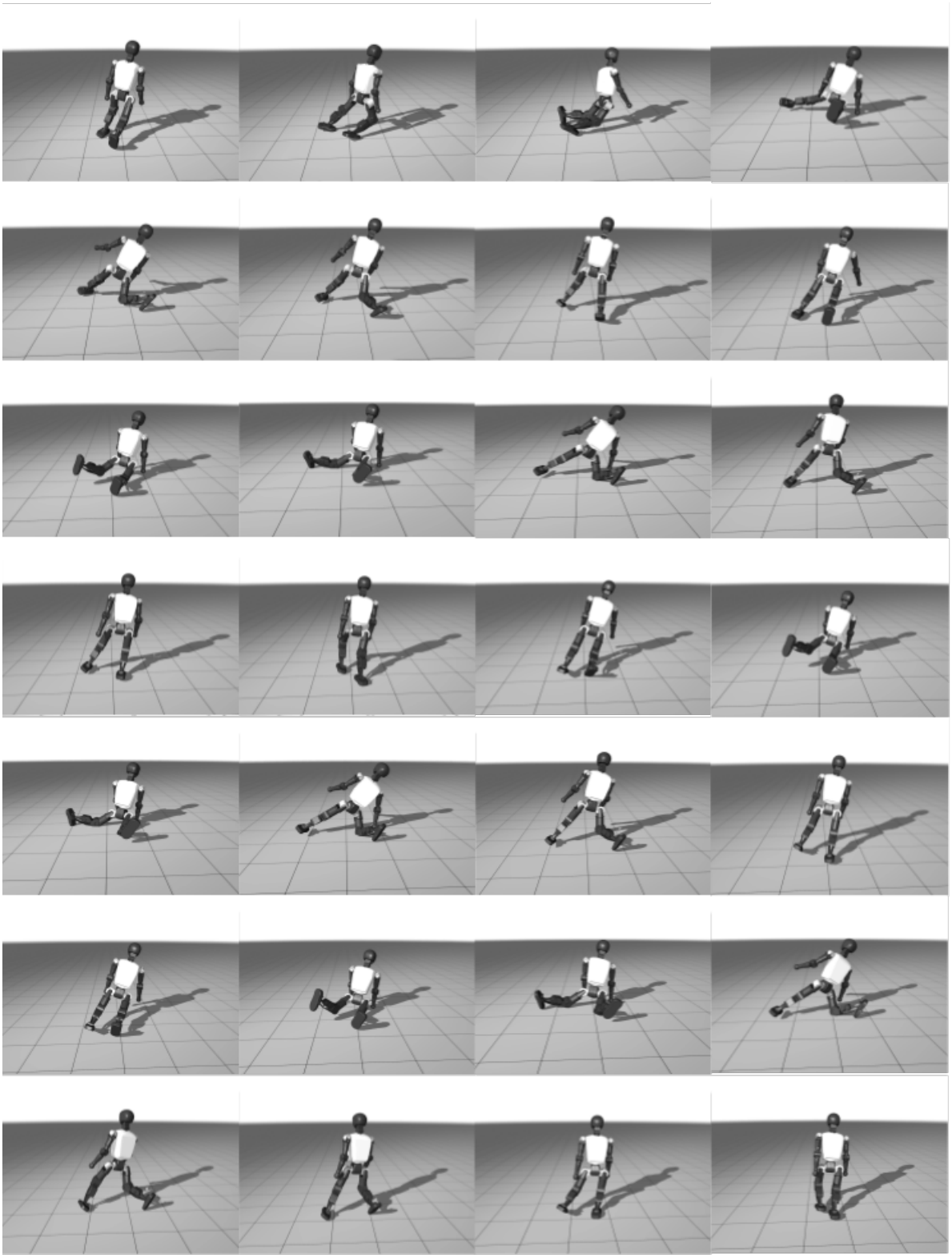

Figure 6.4: (b) Still unable to stand, here are attempts to getup 5-8.

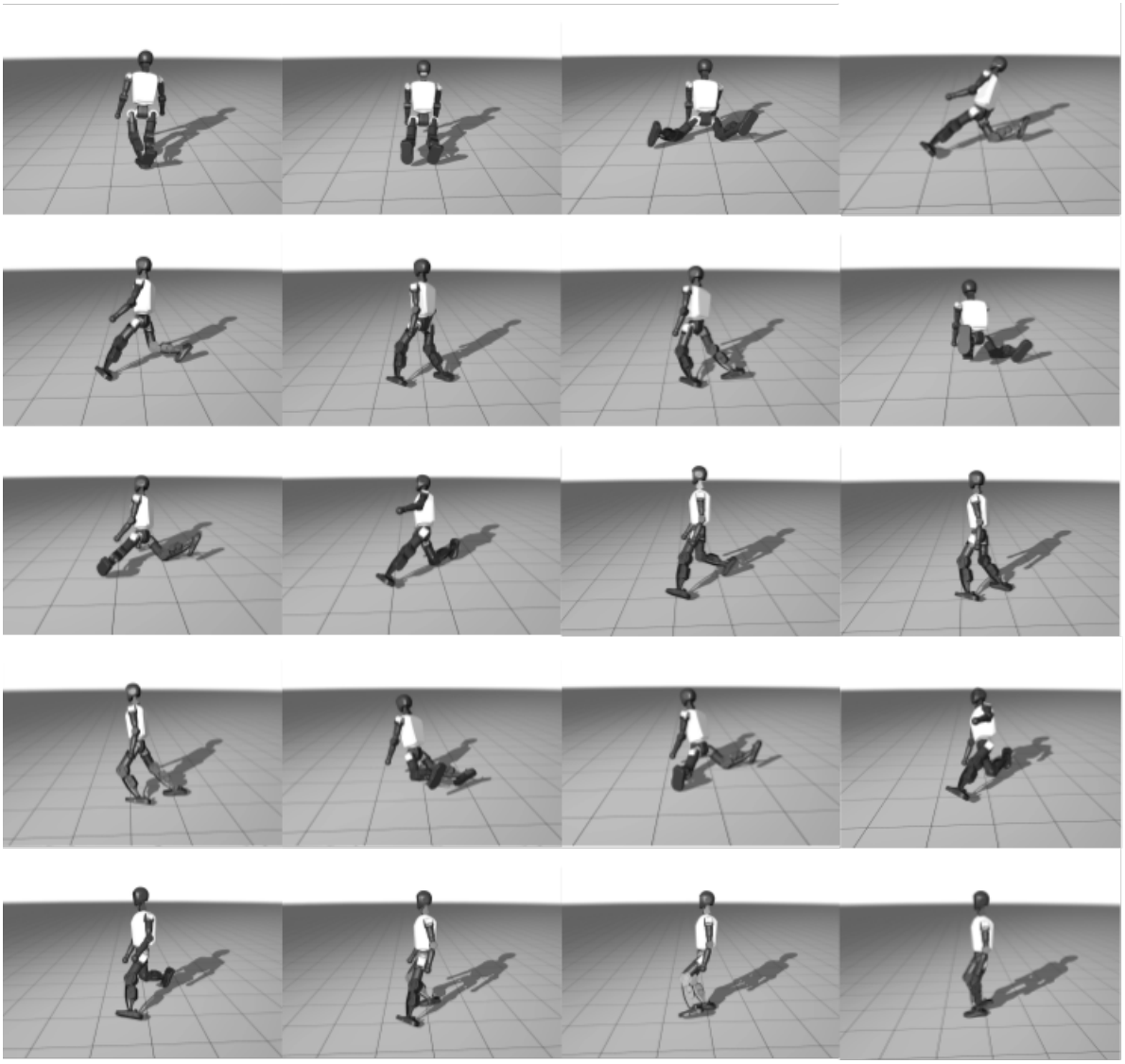

Figure 6.5: (c) Finally the baseline policy succeeds. Shown are attempts 9-11.

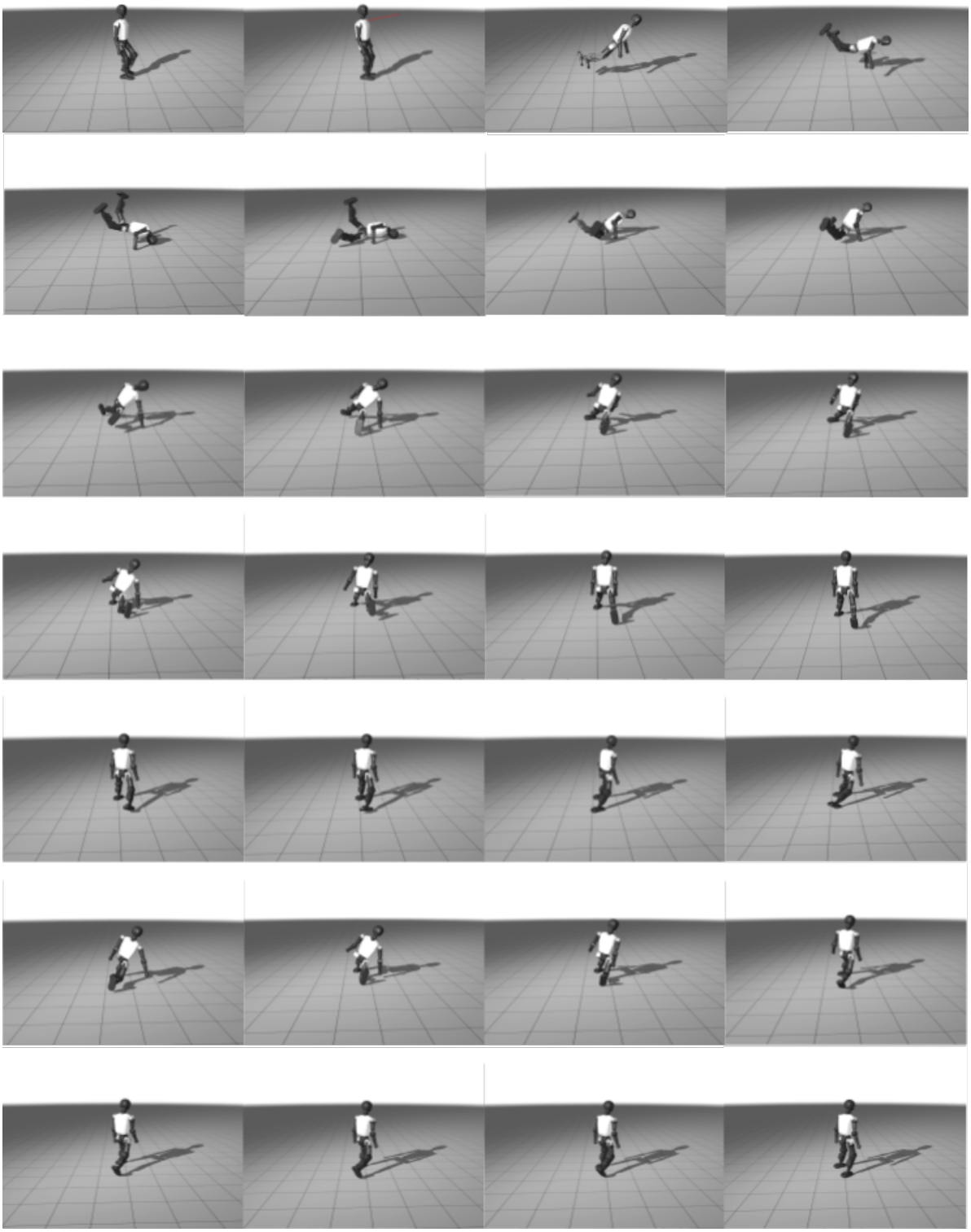

Figure 6.6: The T1 robot is pushed forward from the torso, the push is visualized with the red arrow. DAgger-100k needs two attempts to getup successfully

Table 6.6: Number of getup attempts per trial for each policy (20 trials).

Policy	**Attempts per Trial**																			
Baseline	2	5	1	11	4	10	1	2	3	2	2	2	5	2	3	3	3	1	3	4
DAgger-100k	1	1	1	1	1	1	1	1	1	1	2	1	1	2	1	1	2	1	1	1

To further evaluate the generality and robustness of each policy, we conducted an additional set of 10 backward push trials, in which force impulses were applied to the robot's torso from the front, inducing a backward fall. This scenario simulates cases where the robot might be bumped from the front—e.g., by a human or obstacle—requiring recovery while potentially sliding or tumbling backward. All evaluation rollouts are provided in Appendix D.

As with the forward push experiments, both policies were given no velocity command and were tasked solely with returning to a stable upright stance. The DAgger-100k policy continued to demonstrate strong performance, recovering in all 10 trials with a maximum of only two attempts and an average of 1.1 attempts per trial. The baseline policy, while also successful in all trials, required more attempts on average (1.9), including up to 4 attempts in one case.

Qualitative differences in behavior were also notable. In 3 of the 10 baseline trials, the robot relied on unstable momentum-based maneuvers to complete the getup, including running short distances after standing in order to stabilize. This behavior is undesirable for real-world deployment, as it introduces unpredictable motion and increased risk of collision. In contrast, the DAgger-100k controller completed every recovery without requiring additional locomotion, maintaining in-place, controlled movements throughout.

Table 6.7: Summary of backward push recovery performance over 10 trials.

Metric	**Baseline**	**DAgger-100k**
Average Attempts per Trial	1.9	**1.1**
Maximum Attempts in Any Trial	4	**2**
Minimum Attempts in Any Trial	1	**1**
Trials Requiring Running to Stabilize	3	**0**

Table 6.8: Number of getup attempts per trial for each policy (10 trials).

Policy	**Attempts per Trial**									
Baseline	1	4	2	2	1	2	3	1	1	2
DAgger-100k	1	1	1	1	2	1	1	1	1	1

These additional results further support the conclusion that keyframe-based recovery not only improves energy and torque efficiency but also promotes behavioral stability and spa-

tial discipline—important characteristics for operation in constrained or human-populated environments.

Chapter 7

Conclusion & Discussion

The evaluations presented across the low-wall, puddle, and push environments provide compelling empirical support for the central thesis of this work: that modular, keyframe-based recovery policies, when integrated into a unified end-to-end controller via online DAgger, yield significantly more robust, reliable, and physically grounded behavior than conventional or monolithic learned controllers.

Across all tested environments, the DAgger-100k policy consistently outperformed the baseline in terms of recovery success rate, behavioral stability, and physical efficiency. In every trial—whether involving tripping over obstacles, slipping on low-friction patches, or recovering from external perturbations—the DAgger-100k policy succeeded in regaining a stable standing posture, often with fewer getup attempts and significantly smoother, more controlled motion.

While the DAgger-100k policy occasionally fell more frequently during locomotion—likely due to minor degradation introduced during the student distillation phase—these slips were mitigated by superior recovery capabilities. The controller's ability to consistently succeed across 100% of trials, without requiring excessive or unsafe behavior, reflects a critical advancement toward real-world deployability.

These results substantiate the value of the two primary contributions introduced in this thesis:

- **Keyframe-based recovery decomposition**: Through curriculum-based training, this approach encodes human-inspired recovery strategies into the policy, resulting in physically interpretable, energy-efficient, and mechanically conservative behaviors. As shown in the getup comparison (Table 4.3), the keyframe controller required nearly 30% less energy on average, exhibited significantly lower joint accelerations, and maintained lower maximum torques—all desirable traits for deployment on physical robots.

- **Unified policy integration via DAgger**: The DAgger framework enables smooth unification of locomotion and recovery behaviors into a single reactive controller. It resolves the brittle switching issues typical of hybrid systems, ensuring that transitions

between motion modes (e.g., from walking to falling to recovering) occur seamlessly in response to environmental feedback.

Perhaps most remarkably, the DAgger-100k policy demonstrated emergent anticipatory and corrective behaviors, despite being trained with only simple task objectives and limited action commands. For example, the robot was observed stepping back to avoid falling or re-centering its stance mid-slip—all while executing a fixed forward velocity command. These reflex-like adjustments point to an internalized understanding of stability and physical interaction with the environment, learned implicitly through the training process. Such behavior is particularly critical in dynamic, cluttered, or human-populated spaces, where abrupt or unpredictable movements pose safety risks.

The environments used in this study were deliberately constructed to reflect realistic real-world challenges. Whether navigating low obstacles, walking across slippery terrain, or enduring sudden pushes, the DAgger-100k policy maintained composure, succeeded consistently, and did so with mechanical grace. These are not merely signs of policy success in simulation—they are indicators of real-world readiness.

In sum, this thesis presents a principled and scalable approach to full-body humanoid control. By combining structured keyframe priors with end-to-end learning and online imitation refinement, we show that policies can be made both intelligent and interpretable, robust yet efficient. The work contributes a novel controller architecture that is simultaneously grounded in physical insight and capable of complex, emergent behavior—paving the way for humanoid robots that can walk, fall, and get back up again autonomously, safely, and reliably.

Bibliography

[1] Marcin Andrychowicz et al. *What Matters In On-Policy Reinforcement Learning? A Large-Scale Empirical Study*. 2020. arXiv: 2006.05990 [cs.LG]. URL: https://arxiv.org/abs/2006.05990.

[2] Guillaume Bellegarda and A.J. Ijspeert. "CPG-RL: Learning Central Pattern Generators for Quadruped Locomotion". In: *IEEE Robotics and Automation Letters* PP (Oct. 2022), pp. 1–8. DOI: 10.1109/LRA.2022.3218167.

[3] Yevgen Chebotar et al. *Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience*. 2019. arXiv: 1810.05687 [cs.RO]. URL: https://arxiv.org/abs/1810.05687.

[4] Zixuan Chen et al. "Learning Smooth Humanoid Locomotion Through Lipschitz-Constrained Policies". In: *arXiv preprint arXiv:2410.11825* (2024).

[5] Zixuan Chen et al. *Learning Smooth Humanoid Locomotion through Lipschitz-Constrained Policies*. 2024. arXiv: 2410.11825 [cs.RO]. URL: https://arxiv.org/abs/2410.11825.

[6] Zixuan Chen et al. "Learning Smooth Humanoid Locomotion through Lipschitz-Constrained Policies". In: *arxiv preprint arXiv:2410.11825* (2024).

[7] Xuxin Cheng et al. *Expressive Whole-Body Control for Humanoid Robots*. 2024. arXiv: 2402.16796 [cs.RO]. URL: https://arxiv.org/abs/2402.16796.

[8] Xuxin Cheng et al. *Extreme Parkour with Legged Robots*. 2023. arXiv: 2309.14341 [cs.RO]. URL: https://arxiv.org/abs/2309.14341.

[9] Nuttapong Chentanez et al. "Physics-based motion capture imitation with deep reinforcement learning". In: *Proceedings of the 11th ACM SIGGRAPH Conference on Motion, Interaction and Games*. MIG '18. Limassol, Cyprus: Association for Computing Machinery, 2018. ISBN: 9781450360159. DOI: 10.1145/3274247.3274506. URL: https://doi.org/10.1145/3274247.3274506.

[10] Cheng Chi et al. *Diffusion Policy: Visuomotor Policy Learning via Action Diffusion*. 2024. arXiv: 2303.04137 [cs.RO]. URL: https://arxiv.org/abs/2303.04137.

[11] Matthew Chignoli et al. *The MIT Humanoid Robot: Design, Motion Planning, and Control For Acrobatic Behaviors*. 2021. arXiv: 2104.09025 [cs.RO]. URL: https://arxiv.org/abs/2104.09025.

[12] Paul Christiano et al. *Transfer from Simulation to Real World through Learning Deep Inverse Dynamics Model*. 2016. arXiv: 1610.03518 [cs.RO]. URL: https://arxiv.org/abs/1610.03518.

[13] Erwin Coumans and Yunfei Bai. *Pybullet, a python module for physics simulation for games, robotics and machine learning*. 2016.

[14] Jared Di Carlo et al. "Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 1–9. DOI: 10.1109/IROS.2018.8594448.

[15] Pranay Dugar et al. *Learning Multi-Modal Whole-Body Control for Real-World Humanoid Robots*. 2024. arXiv: 2408.07295 [cs.RO]. URL: https://arxiv.org/abs/2408.07295.

[16] Yanan Fan et al. "Quadruped robot locomotion via soft actor-critic with muti-head critic and dynamic policy gradient". In: *Applied Intelligence* 55 (May 2025). DOI: 10.1007/s10489-025-06584-1.

[17] Gilbert Feng et al. "GenLoco: Generalized Locomotion Controllers for Quadrupedal Robots". In: *Conference on Robot Learning (CoRL)*. PMLR, 2023, pp. 1893–1903.

[18] A. Frezzato, A. Tangri, and Sheldon Andrews. "Synthesizing Get-Up Motions for Physics-based Characters". In: *Computer Graphics Forum* 41 (Mar. 2023), pp. 207–218. DOI: 10.1111/cgf.14636.

[19] Zipeng Fu, Xuxin Cheng, and Deepak Pathak. "Deep Whole-Body Control: Learning a Unified Policy for Manipulation and Locomotion". In: *Proceedings of The 6th Conference on Robot Learning*. Ed. by Karen Liu, Dana Kulic, and Jeff Ichnowski. Vol. 205. Proceedings of Machine Learning Research. PMLR, 2023, pp. 138–149. URL: https://proceedings.mlr.press/v205/fu23a.html.

[20] Zipeng Fu et al. *Minimizing Energy Consumption Leads to the Emergence of Gaits in Legged Robots*. 2021. arXiv: 2111.01674 [cs.RO]. URL: https://arxiv.org/abs/2111.01674.

[21] Zipeng Fu et al. "Minimizing Energy Consumption Leads to the Emergence of Gaits in Legged Robots". In: *Proceedings of the 5th Conference on Robot Learning*. Ed. by Aleksandra Faust, David Hsu, and Gerhard Neumann. Vol. 164. Proceedings of Machine Learning Research. PMLR, 2022, pp. 928–937. URL: https://proceedings.mlr.press/v164/fu22a.html.

[22] Kiyoshi Fujiwara et al. "UKEMI: Falling motion control to minimize damage to biped humanoid robot". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2002)*. Vol. 3. 2002, pp. 2521–2526. ISBN: 0-7803-7398-7. URL: http://staff.aist.go.jp/k.kaneko/publications/2002_publications/02-9.pdf.

[23] Clément Gaspard et al. "FRASA: An End-to-End Reinforcement Learning Agent for Fall Recovery and Stand-Up of Humanoid Robots". In: *arXiv preprint arXiv:2410.08655* (2024).

[24] Miguel González-Fierro et al. "A humanoid robot standing up through learning from demonstration using a multimodal reward function". In: *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. 2013, pp. 74–79. DOI: 10.1109/HUMANOIDS.2013.7029958.

[25] Jessy Grizzle et al. "MABEL, a new robotic bipedal walker and runner". In: *2009 American Control Conf.* (2009). DOI: 10.1109/ACC.2009.5160550.

[26] Zhaoyuan Gu et al. *Humanoid Locomotion and Manipulation: Current Progress and Challenges in Control, Planning, and Learning*. Jan. 2025. DOI: 10.48550/arXiv.2501.02116.

[27] Tuomas Haarnoja et al. *Learning to Walk via Deep Reinforcement Learning*. 2019. arXiv: 1812.11103 [cs.LG]. URL: https://arxiv.org/abs/1812.11103.

[28] Tuomas Haarnoja et al. *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor*. Jan. 2018. DOI: 10.48550/arXiv.1801.01290.

[29] Jiang Han et al. "An energy-efficient obstacle-crossing control framework for quadruped robots". In: *Results in Engineering* 26 (Mar. 2025), p. 104661. DOI: 10.1016/j.rineng.2025.104661.

[30] Ryan Hoque et al. *Fleet-DAgger: Interactive Robot Fleet Learning with Scalable Human Supervision*. 2022. arXiv: 2206.14349 [cs.RO]. URL: https://arxiv.org/abs/2206.14349.

[31] Ryan Hoque et al. *ThriftyDAgger: Budget-Aware Novelty and Risk Gating for Interactive Imitation Learning*. 2021. arXiv: 2109.08273 [cs.RO]. URL: https://arxiv.org/abs/2109.08273.

[32] Jemin Hwangbo et al. "Learning agile and dynamic motor skills for legged robots". In: *Science Robotics* 4.26 (2019), eaau5872. DOI: 10.1126/scirobotics.aau5872. eprint: https://www.science.org/doi/pdf/10.1126/scirobotics.aau5872. URL: https://www.science.org/doi/abs/10.1126/scirobotics.aau5872.

[33] Jemin Hwangbo et al. "Learning agile and dynamic motor skills for legged robots". In: *Science Robotics* 4.26 (Jan. 2019). ISSN: 2470-9476. DOI: 10.1126/scirobotics.aau5872. URL: http://dx.doi.org/10.1126/scirobotics.aau5872.

[34] Julian Ibarz et al. "How to train your robot with deep reinforcement learning: lessons we have learned". In: *The International Journal of Robotics Research* 40.4–5 (Jan. 2021), pp. 698–721. ISSN: 1741-3176. DOI: 10.1177/0278364920987859. URL: http://dx.doi.org/10.1177/0278364920987859.

[35] Atil Iscen et al. "Learning Agile Locomotion Skills with a Mentor". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. Xi'an, China: IEEE Press, 2021. DOI: 10.1109/ICRA48506.2021.9561567. URL: https://doi.org/10.1109/ICRA48506.2021.9561567.

[36] Manuel Kaspar, Juan David Munoz Osorio, and Jürgen Bock. *Sim2Real Transfer for Reinforcement Learning without Dynamics Randomization*. 2020. arXiv: 2002.11635 [cs.AI]. URL: https://arxiv.org/abs/2002.11635.

[37] Michael Kelly et al. *HG-DAgger: Interactive Imitation Learning with Human Experts*. 2019. arXiv: 1810.02890 [cs.RO]. URL: https://arxiv.org/abs/1810.02890.

[38] N. Kohl and Peter Stone. "Policy gradient reinforcement learning for fast quadrupedal locomotion". In: *IEEE International Conference on Robotics and Automation* (2004).

[39] Eric Krotkov et al. "The DARPA Robotics Challenge Finals: Results and Perspectives". In: *J. Field Robot.* 34.2 (Mar. 2017), pp. 229–240. ISSN: 1556-4959. DOI: 10.1002/rob.21683. URL: https://doi.org/10.1002/rob.21683.

[40] Scott Kuindersma et al. "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot". In: *Auton. Robots* 40.3 (Mar. 2016), pp. 429–455. ISSN: 0929-5593. DOI: 10.1007/s10514-015-9479-3. URL: https://doi.org/10.1007/s10514-015-9479-3.

[41] Ashish Kumar et al. *RMA: Rapid Motor Adaptation for Legged Robots*. 2021. arXiv: 2107.04034 [cs.LG]. URL: https://arxiv.org/abs/2107.04034.

[42] Joonho Lee et al. "Learning quadrupedal locomotion over challenging terrain". In: *Science Robotics* 5.47 (Oct. 2020). ISSN: 2470-9476. DOI: 10.1126/scirobotics.abc5986. URL: http://dx.doi.org/10.1126/scirobotics.abc5986.

[43] Sung-Wook Lee, Xuhui Kang, and Yen-Ling Kuo. *Diff-DAgger: Uncertainty Estimation with Diffusion Policy for Robotic Manipulation*. 2025. arXiv: 2410.14868 [cs.RO]. URL: https://arxiv.org/abs/2410.14868.

[44] Zhongyu Li et al. *Robust and Versatile Bipedal Jumping Control through Reinforcement Learning*. 2023. arXiv: 2302.09450 [cs.RO]. URL: https://arxiv.org/abs/2302.09450.

[45] Zhongyu Li et al. "Robust and Versatile Bipedal Jumping Control through Reinforcement Learning". In: *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*. Ed. by Kostas E. Bekris et al. 2023. DOI: 10.15607/RSS.2023.XIX.052. URL: https://doi.org/10.15607/RSS.2023.XIX.052.

[46] Tyler Ga Wei Lum et al. *DextrAH-G: Pixels-to-Action Dexterous Arm-Hand Grasping with Geometric Fabrics*. 2024. arXiv: 2407.02274 [cs.RO]. URL: https://arxiv.org/abs/2407.02274.

[47] Viktor Makoviychuk et al. *Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning*. 2021. arXiv: 2108.10470 [cs.RO]. URL: https://arxiv.org/abs/2108.10470.

[48] Ajay Mandlekar et al. *What Matters in Learning from Offline Human Demonstrations for Robot Manipulation*. 2021. arXiv: 2108.03298 [cs.RO]. URL: https://arxiv.org/abs/2108.03298.

[49] Ajay Mandlekar et al. *What Matters in Learning from Offline Human Demonstrations for Robot Manipulation*. 2021. arXiv: 2108.03298 [cs.RO]. URL: https://arxiv.org/abs/2108.03298.

[50] Josh Merel et al. *Hierarchical visuomotor control of humanoids*. 2019. arXiv: 1811.09656 [cs.AI]. URL: https://arxiv.org/abs/1811.09656.

[51] Muhammad Sunny Nazeer, Cecilia Laschi, and Egidio Falotico. "Soft DAgger: Sample-Efficient Imitation Learning for Control of Soft Robots". In: *Sensors* 23.19 (2023), p. 8278. DOI: 10.3390/s23198278. URL: https://www.mdpi.com/1424-8220/23/19/8278.

[52] OpenAI et al. *Learning Dexterous In-Hand Manipulation*. 2019. arXiv: 1808.00177 [cs.LG]. URL: https://arxiv.org/abs/1808.00177.

[53] Xue Bin Peng, Glen Berseth, and Michiel van de Panne. "Terrain-adaptive locomotion skills using deep reinforcement learning". In: *ACM Trans. Graph.* 35.4 (July 2016). ISSN: 0730-0301. DOI: 10.1145/2897824.2925881. URL: https://doi.org/10.1145/2897824.2925881.

[54] Xue Bin Peng et al. *AMP: adversarial motion priors for stylized physics-based character control*. New York, NY, USA, 2021. DOI: 10.1145/3450626.3459670. URL: https://doi.org/10.1145/3450626.3459670.

[55] Xue Bin Peng et al. "DeepLoco: Dynamic Locomotion Skills Using Hierarchical Deep Reinforcement Learning". In: *ACM Trans. Graph.* 36.4 (July 2017), 41:1–41:13. ISSN: 0730-0301. DOI: 10.1145/3072959.3073602. URL: http://doi.acm.org/10.1145/3072959.3073602.

[56] Xue Bin Peng et al. "DeepMimic: example-guided deep reinforcement learning of physics-based character skills". In: *ACM Trans. Graph.* 37.4 (July 2018). ISSN: 0730-0301. DOI: 10.1145/3197517.3201311. URL: https://doi.org/10.1145/3197517.3201311.

[57] XingCheng Pu and Congde Zhang. *Decentralized Policy Learning for Quadruped Robots Multi-Gait Control*. Mar. 2023. DOI: 10.21203/rs.3.rs-2687233/v1.

[58] Stephane Ross, Geoffrey J Gordon, and J Andrew Bagnell. "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning". In: *arXiv preprint arXiv:1011.0686* (2011).

[59] Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. *A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning*. 2011. arXiv: 1011.0686 [cs.LG]. URL: https://arxiv.org/abs/1011.0686.

[60] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347 [cs.LG]. URL: https://arxiv.org/abs/1707.06347.

[61] Laura Smith, Yunhao Cao, and Sergey Levine. *Grow Your Limits: Continuous Improvement with Real-World RL for Robotic Locomotion*. 2023. arXiv: 2310.17634 [cs.RO]. URL: https://arxiv.org/abs/2310.17634.

[62] Laura Smith, Ilya Kostrikov, and Sergey Levine. *A Walk in the Park: Learning to Walk in 20 Minutes With Model-Free Reinforcement Learning*. 2022. arXiv: 2208.07860 [cs.RO]. URL: https://arxiv.org/abs/2208.07860.

[63] Laura Smith et al. *Learning and Adapting Agile Locomotion Skills by Transferring Experience*. 2023. arXiv: 2304.09834 [cs.RO]. URL: https://arxiv.org/abs/2304.09834.

[64] Laura Smith et al. "Legged Robots that Keep on Learning: Fine-Tuning Locomotion Policies in the Real World". In: *2022 International Conference on Robotics and Automation (ICRA)*. Philadelphia, PA, USA: IEEE Press, 2022, pp. 1593–1599. DOI: 10.1109/ICRA46639.2022.9812166. URL: https://doi.org/10.1109/ICRA46639.2022.9812166.

[65] Sebastian Stelter et al. "Fast and Reliable Stand-Up Motions for Humanoid Robots Using Spline Interpolation and Parameter Optimization". In: *2021 20th International Conference on Advanced Robotics (ICAR)* (2021), pp. 253–260. URL: https://api.semanticscholar.org/CorpusID:245706908.

[66] Jie Tan et al. *Sim-to-Real: Learning Agile Locomotion For Quadruped Robots*. June 2018. DOI: 10.15607/RSS.2018.XIV.010.

[67] Jie Tan et al. *Sim-to-Real: Learning Agile Locomotion For Quadruped Robots*. 2018. arXiv: 1804.10332 [cs.RO]. URL: https://arxiv.org/abs/1804.10332.

[68] Chen Tang et al. *Deep Reinforcement Learning for Robotics: A Survey of Real-World Successes*. 2024. arXiv: 2408.03539 [cs.RO]. URL: https://arxiv.org/abs/2408.03539.

[69] Josh Tobin et al. "Domain randomization for transferring deep neural networks from simulation to the real world". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vancouver, BC, Canada: IEEE Press, 2017, pp. 23–30. DOI: 10.1109/IROS.2017.8202133. URL: https://doi.org/10.1109/IROS.2017.8202133.

[70] Emanuel Todorov, Tom Erez, and Yuval Tassa. "MuJoCo: A physics engine for model-based control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 5026–5033. DOI: 10.1109/IROS.2012.6386109.

[71] Unitree Robotics. *Unitree G1: Humanoid Agent AI Avatar*. Accessed: 2025-05-10. 2024. URL: https://www.unitree.com/g1.

[72] Yuliu Wang, Ryusuke Sagawa, and Yusuke Yoshiyasu. "Learning Advanced Locomotion for Quadrupedal Robots: A Distributed Multi-Agent Reinforcement Learning Framework with Riemannian Motion Policies". In: *Robotics* 13 (May 2024), p. 86. DOI: 10.3390/robotics13060086.

[73] Kevin Zakka et al. *MuJoCo Playground*. 2025. arXiv: 2502.08844 [cs.RO]. URL: https://arxiv.org/abs/2502.08844.

[74] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. "Sim-to-real transfer in deep reinforcement learning for robotics: a survey". In: *2020 IEEE symposium series on computational intelligence (SSCI)*. IEEE. 2020, pp. 737–744.

[75] Ziwen Zhuang, Shenzhe Yao, and Hang Zhao. *Humanoid Parkour Learning*. 2024. arXiv: 2406.10759 [cs.RO]. URL: https://arxiv.org/abs/2406.10759.

Appendix

Appendix A

Timed Wall Evaluation Trials

DAgger-100k Wall Trials

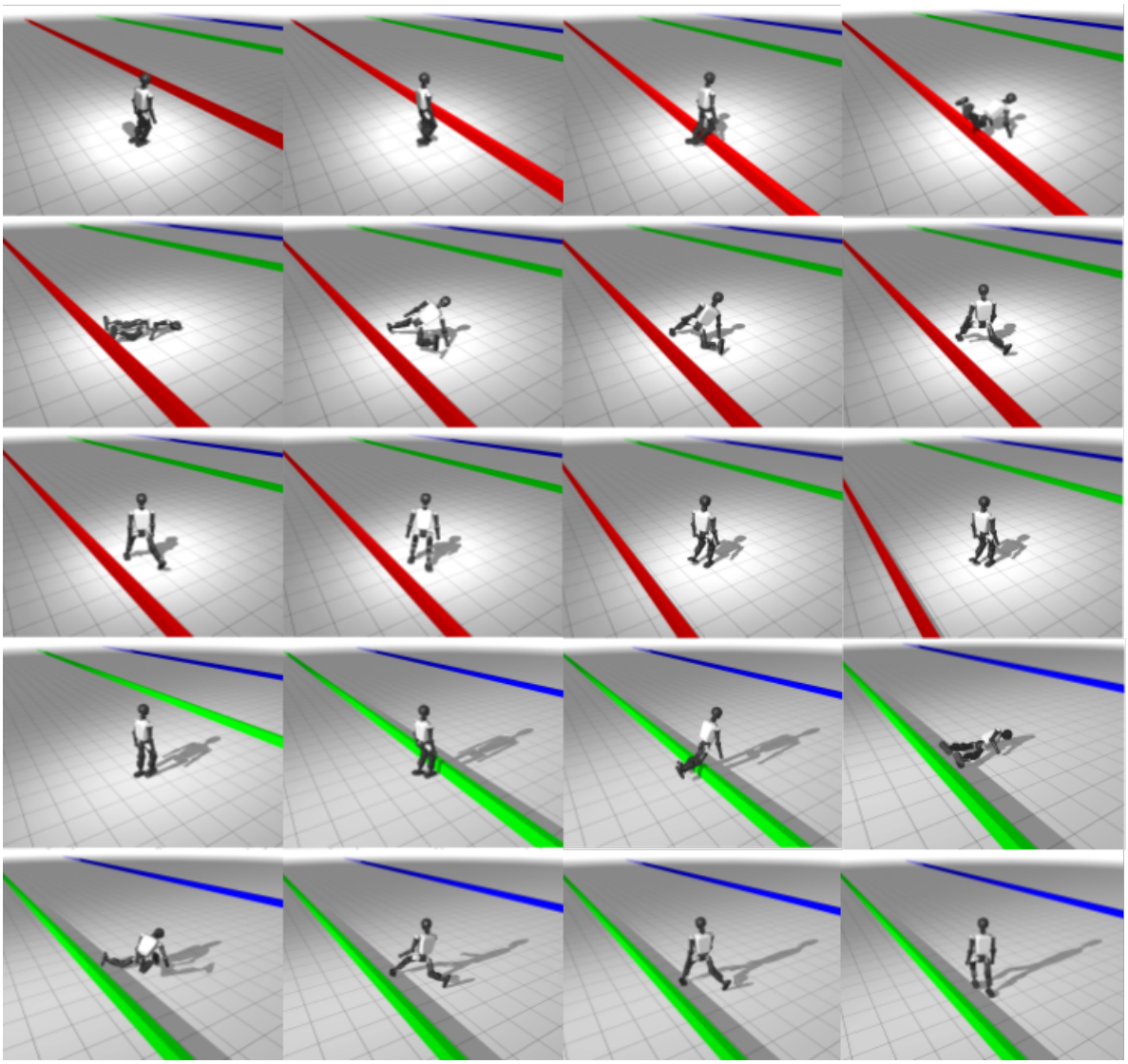

Figure A.1: (a) DAgger-100k timed wall trial 1.

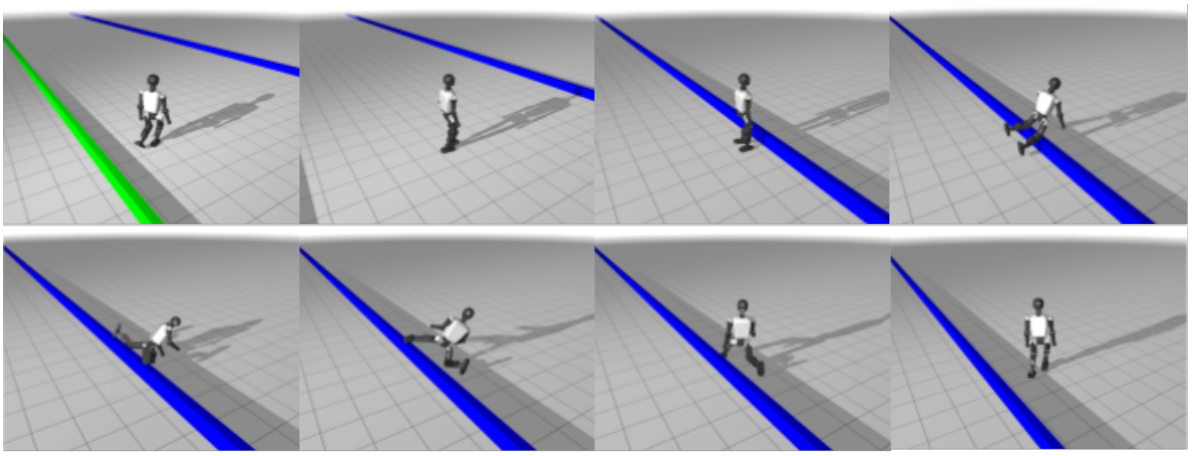

Figure A.2: (b) DAgger-100k timed wall trial 1 cont.

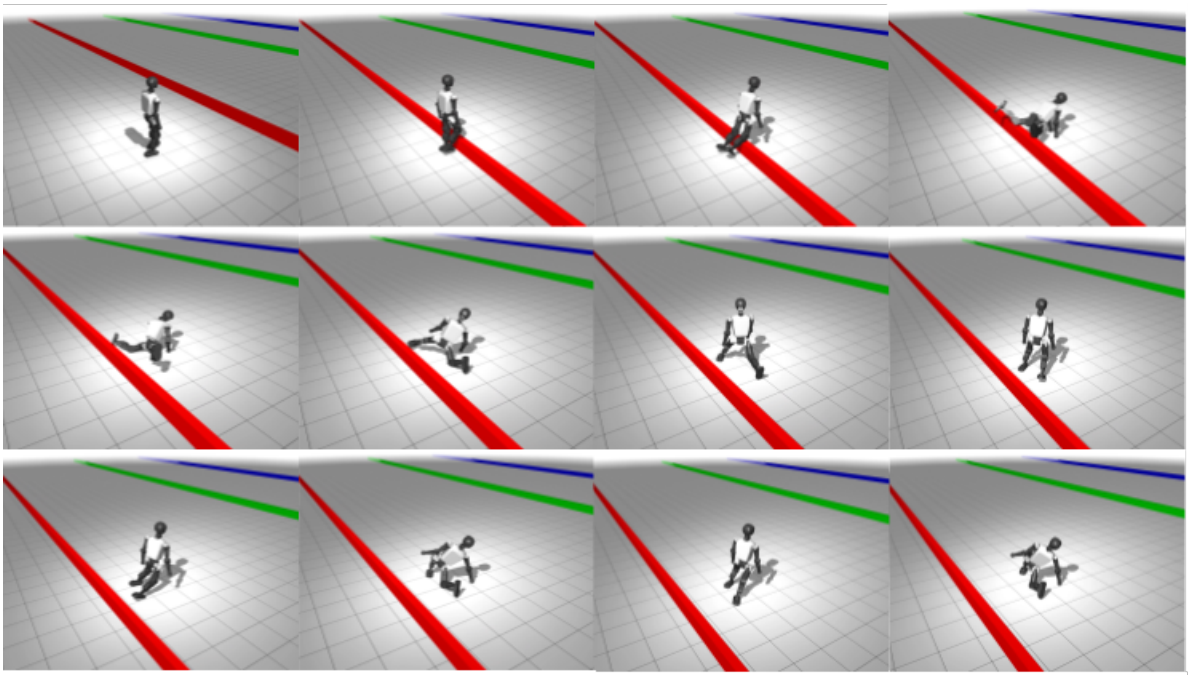

Figure A.3: (a) DAgger-100k timed wall trial 2

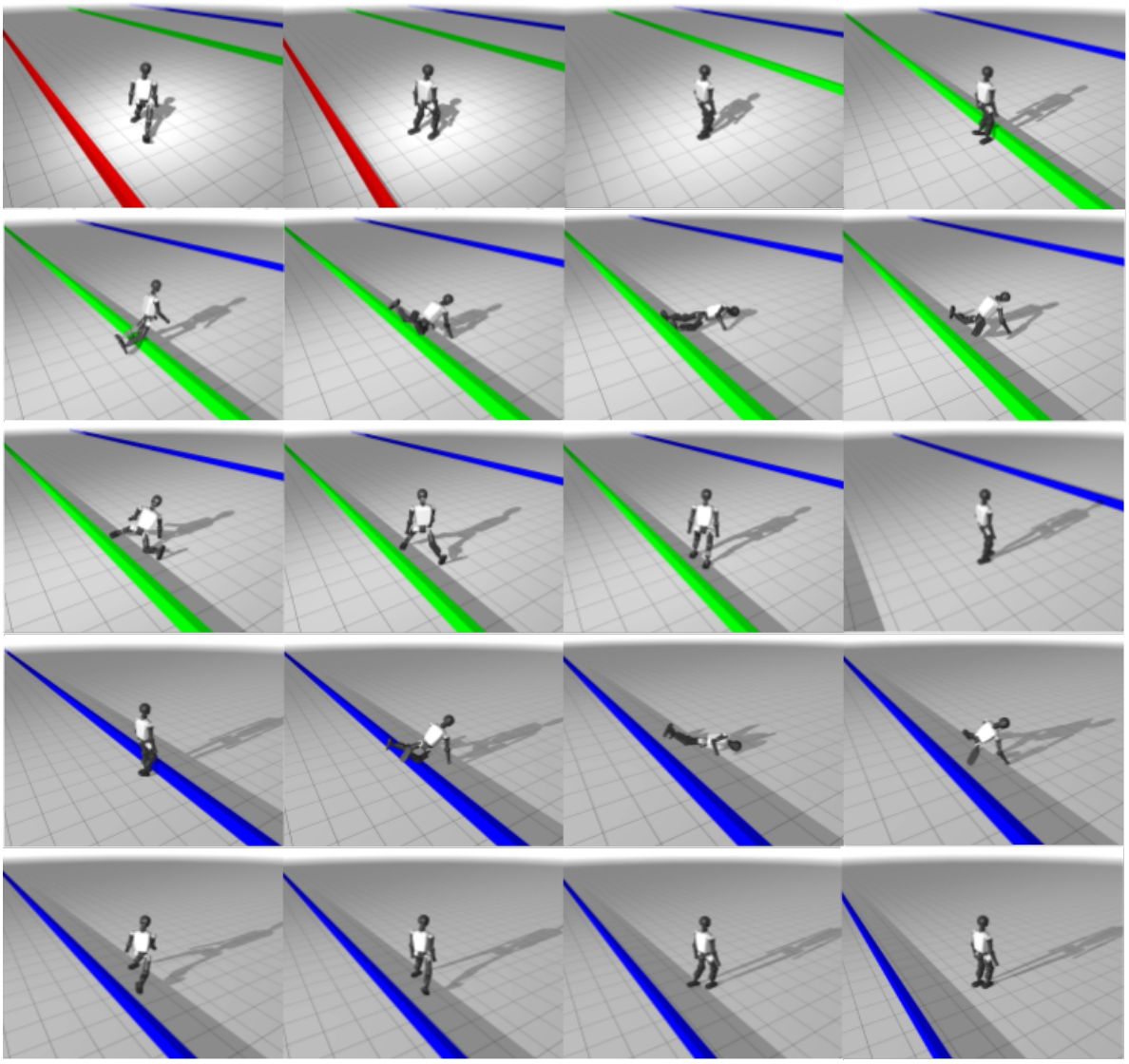

Figure A.4: (b) DAgger-100k timed wall trial 2

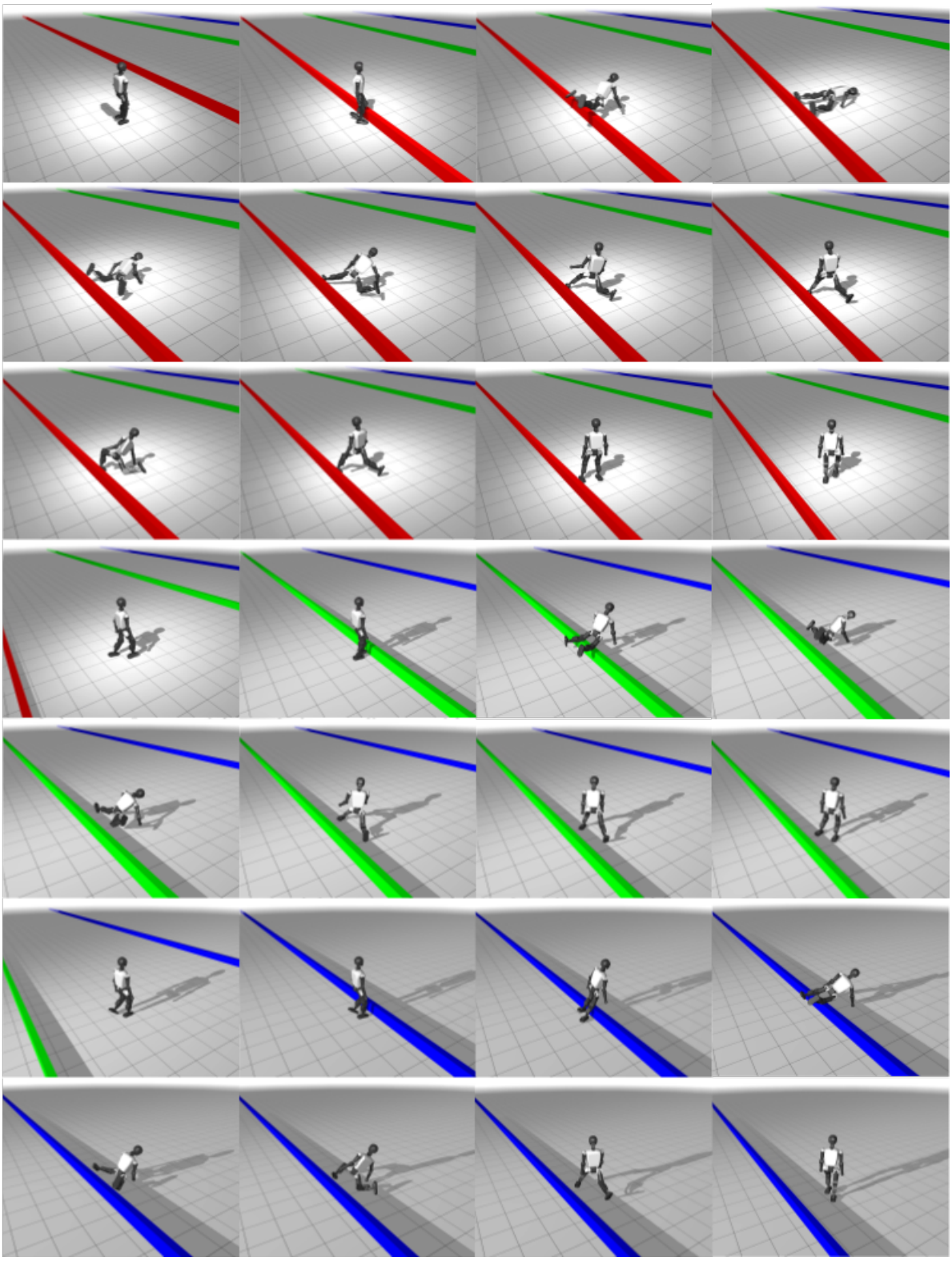

Figure A.5: DAgger-100k timed wall trial 3

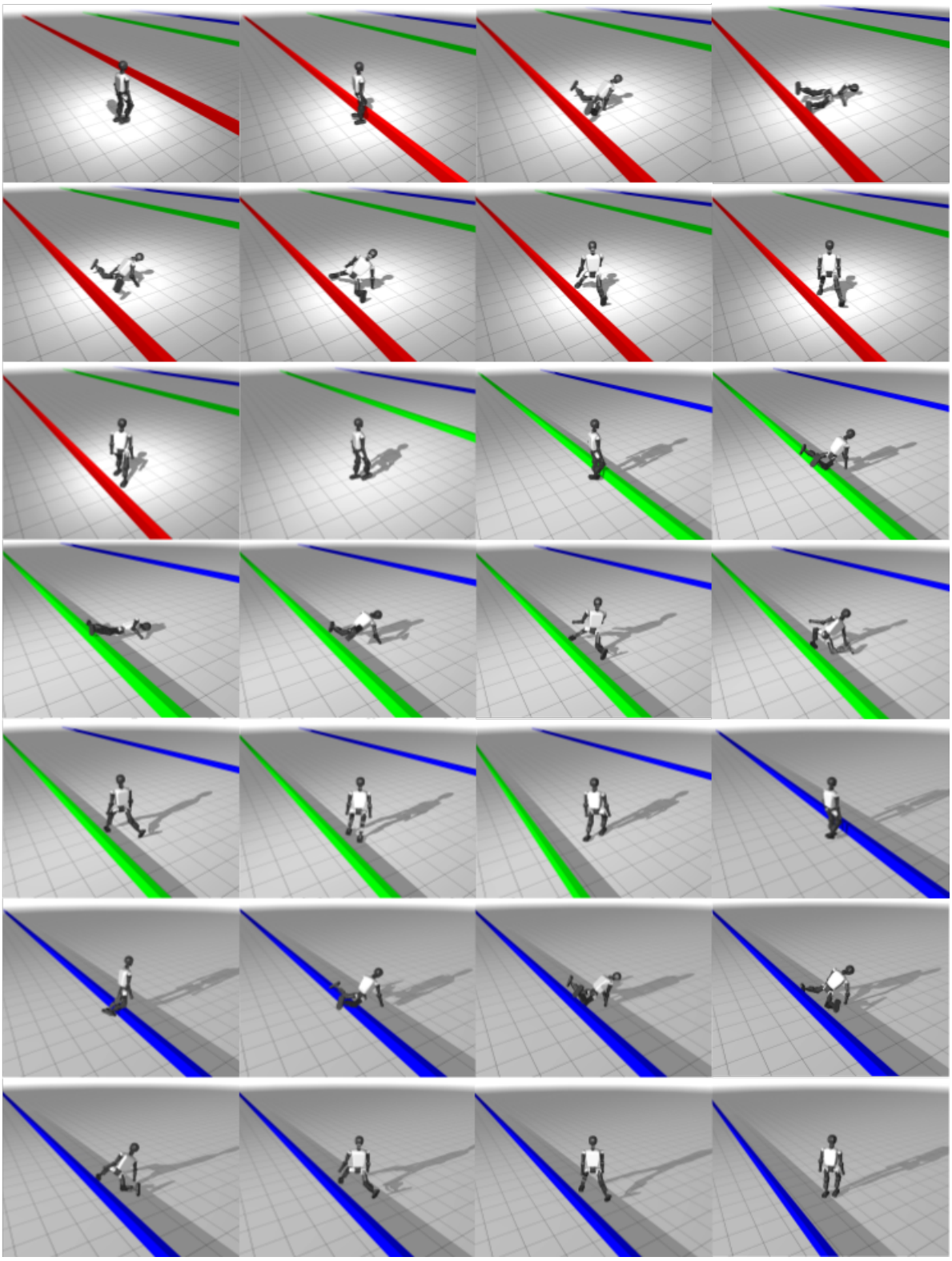

Figure A.6: DAgger-100k timed wall trial 4

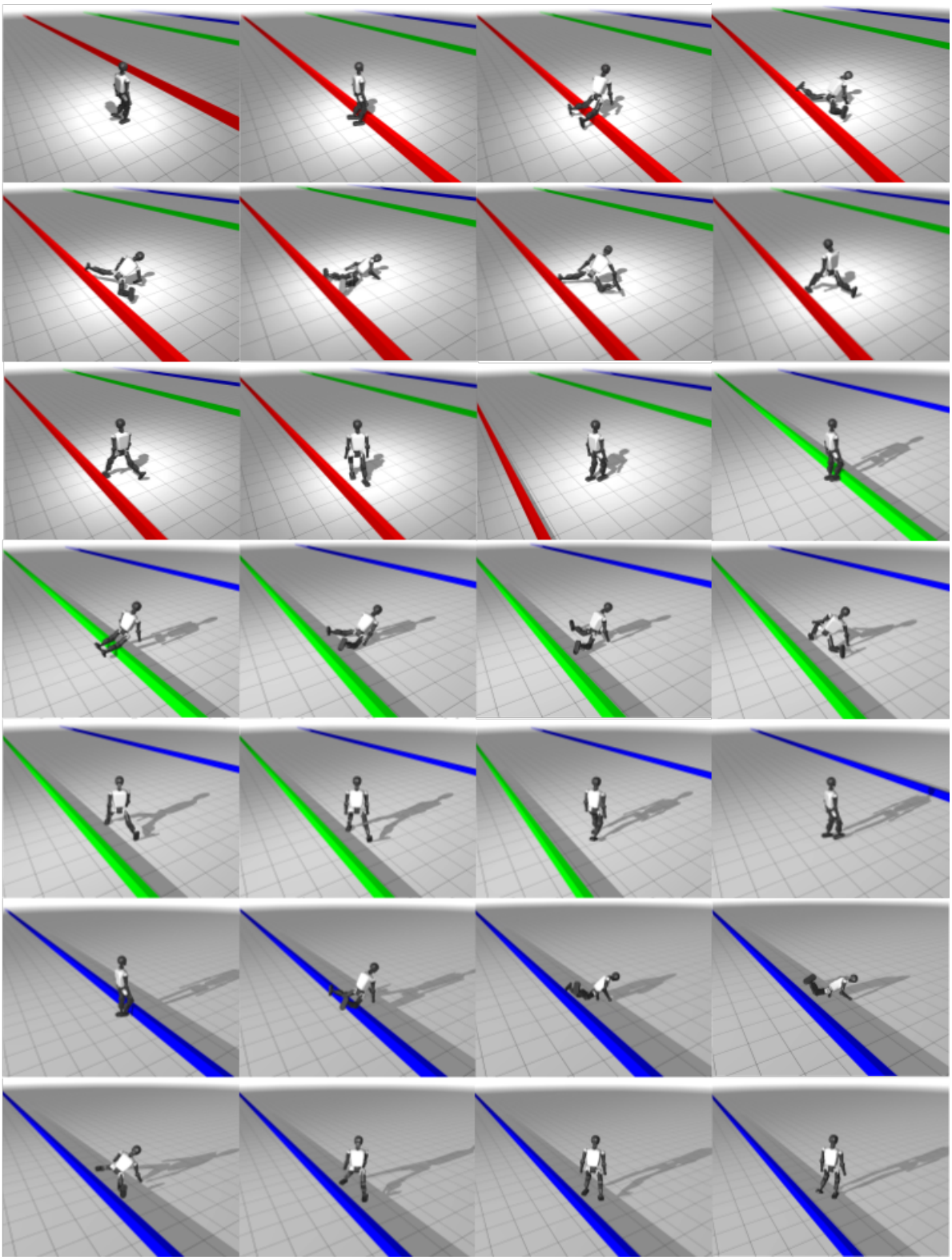

Figure A.7: DAgger-100k timed wall trial 5

Baseline Wall Trials

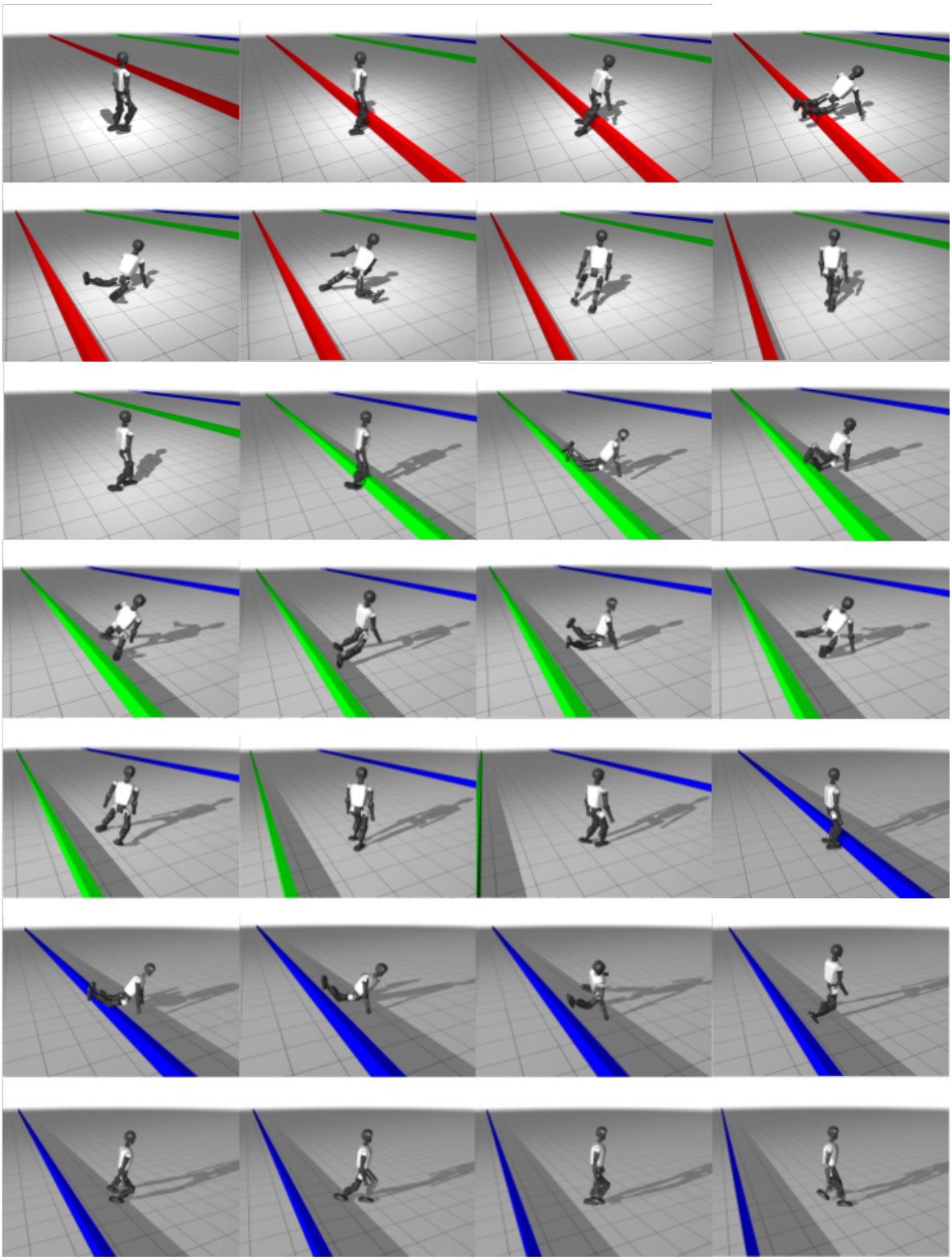

Figure A.8: Baseline timed wall trial 1

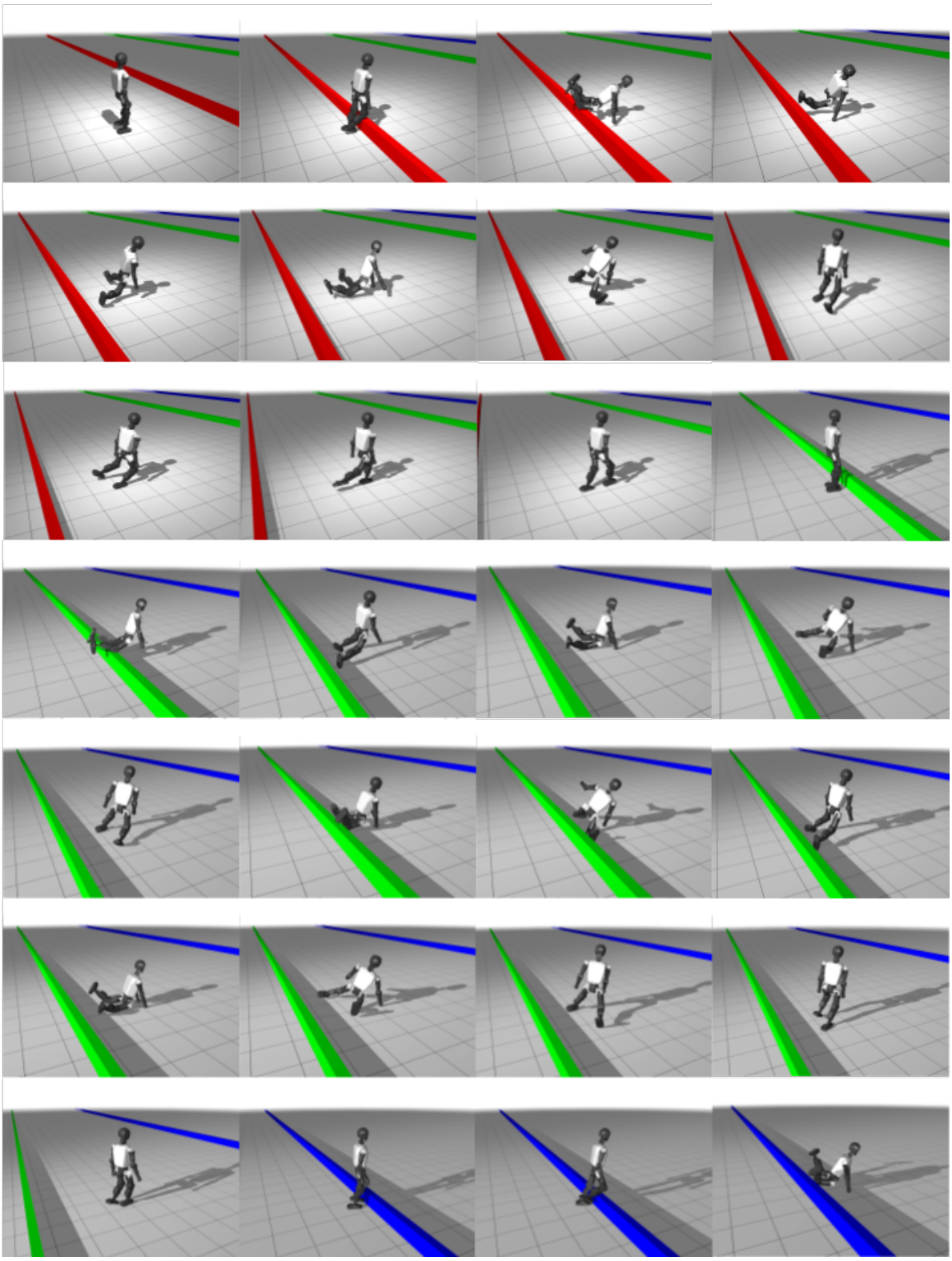

Figure A.9: (a) Baseline timed wall trial 2

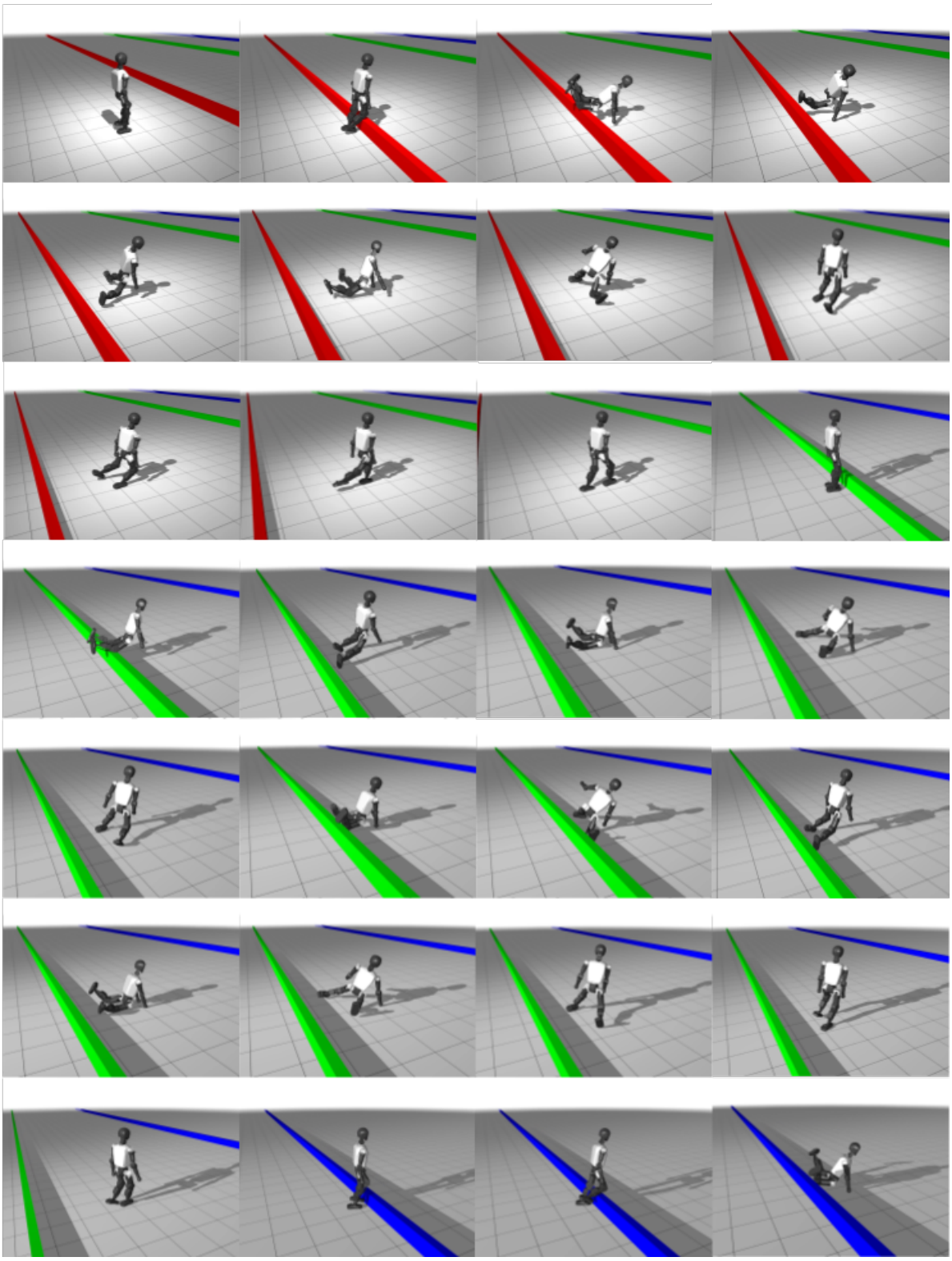

Figure A.10: (a) Baseline timed wall trial 2

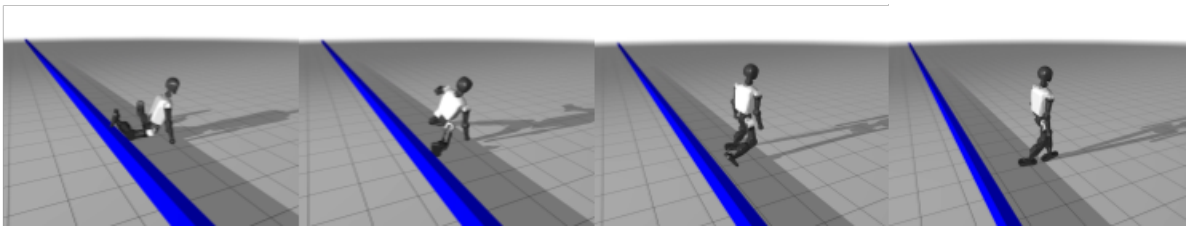

Figure A.11: (b) Baseline timed wall trial 2

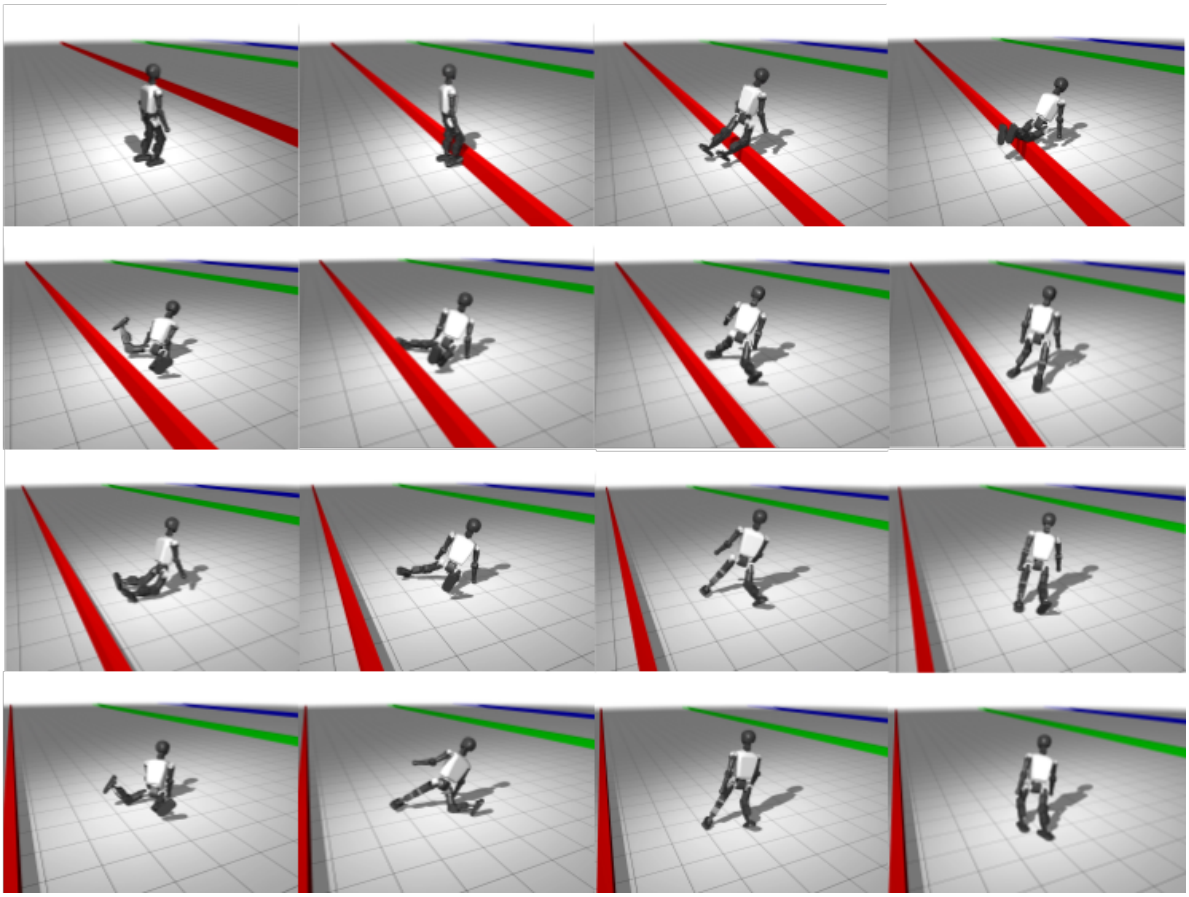

Figure A.12: (a) Baseline timed wall trial 3

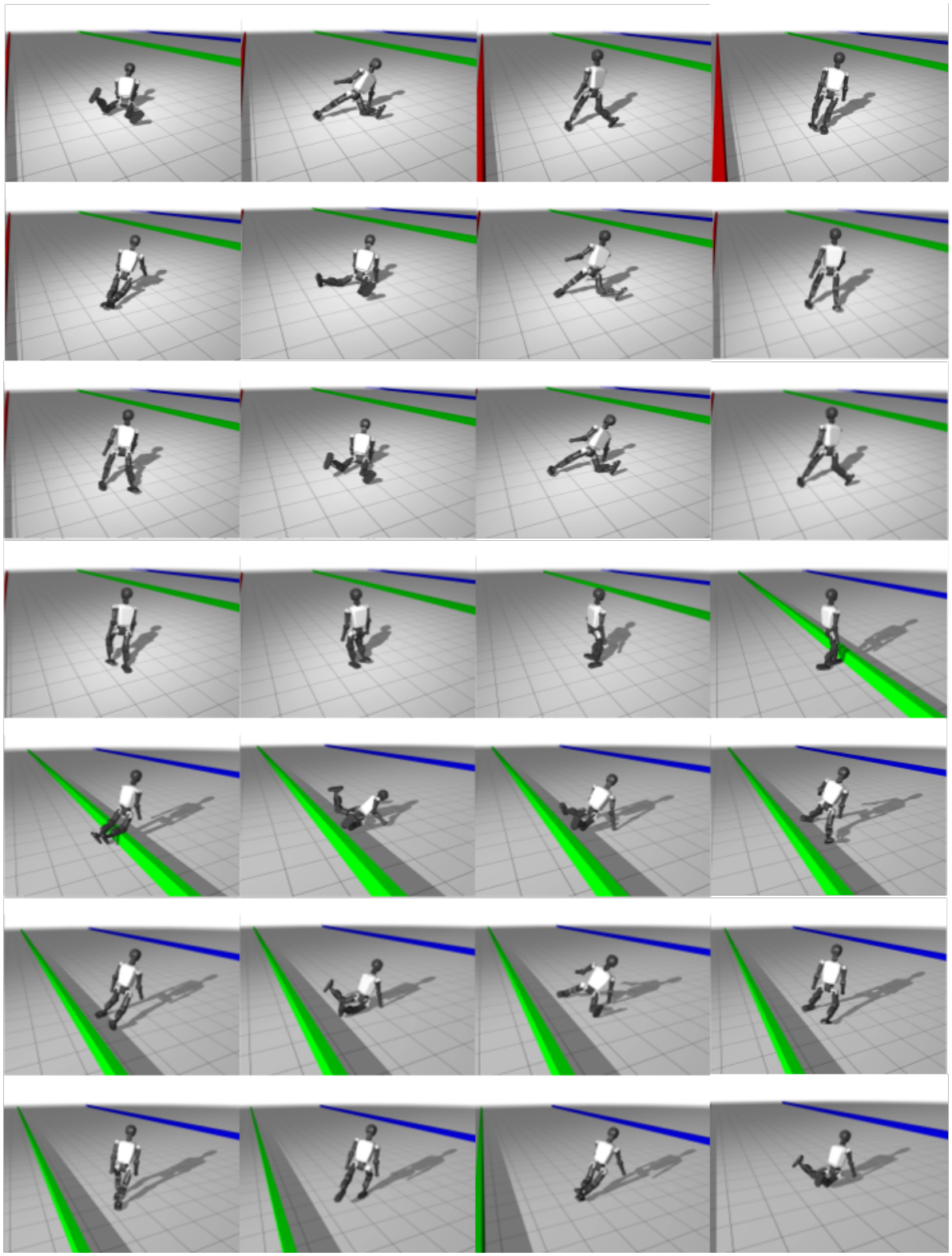

Figure A.13: (b) Baseline timed wall trial 3

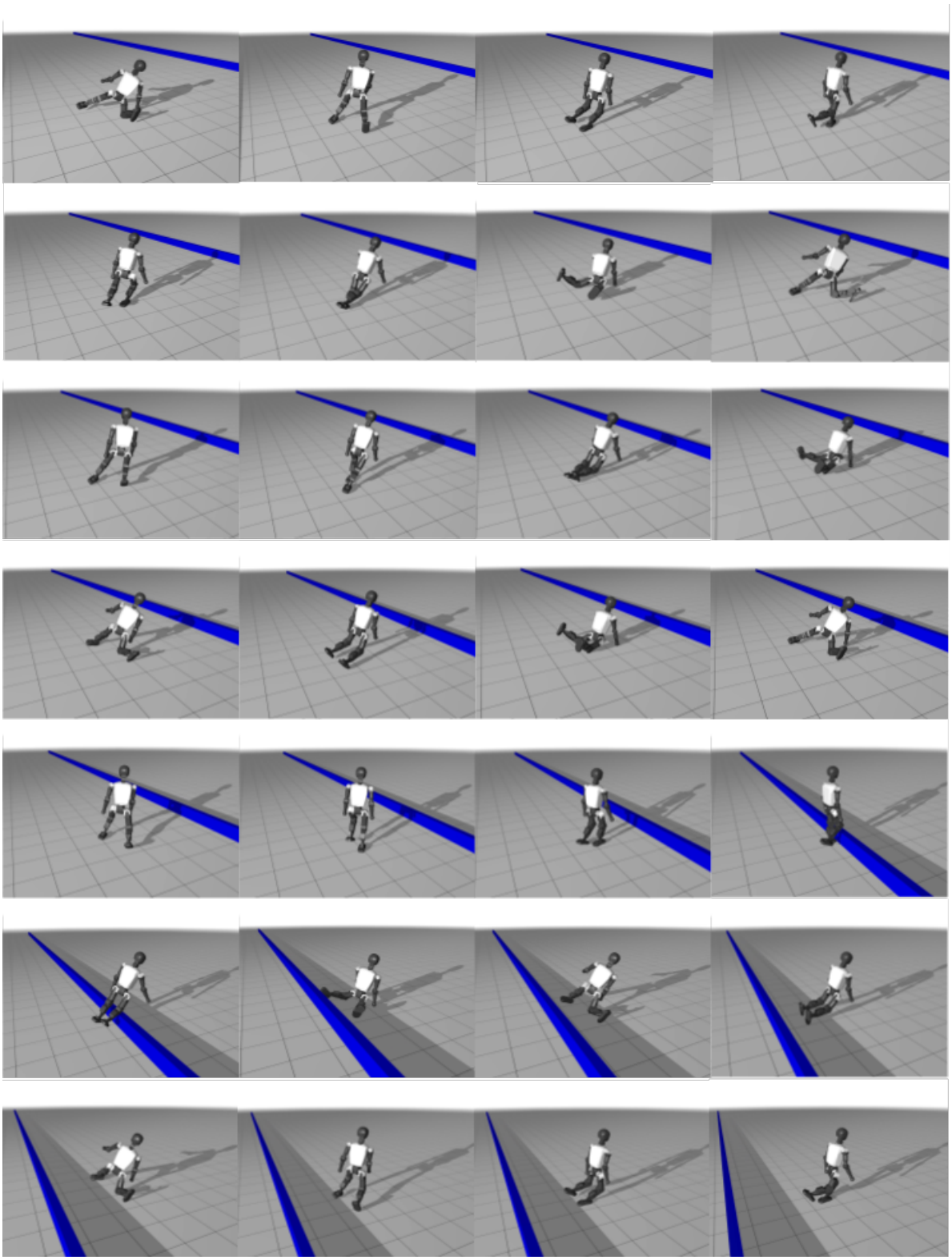

Figure A.14: (c) Baseline timed wall trial 3

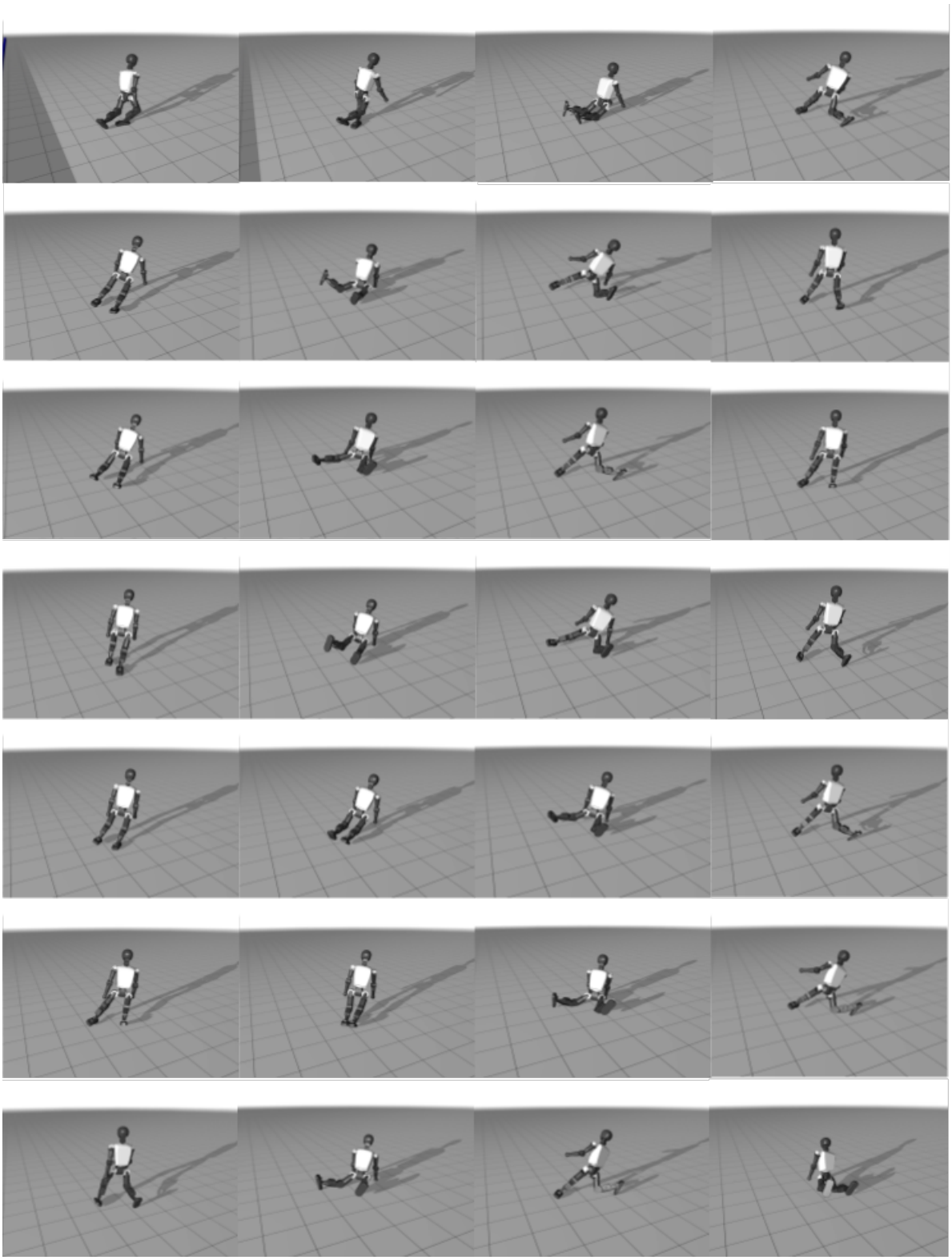

Figure A.15: (d) Baseline timed wall trial 3. The robot gets stuck in an endless loop of falling down and attempting to get back up.

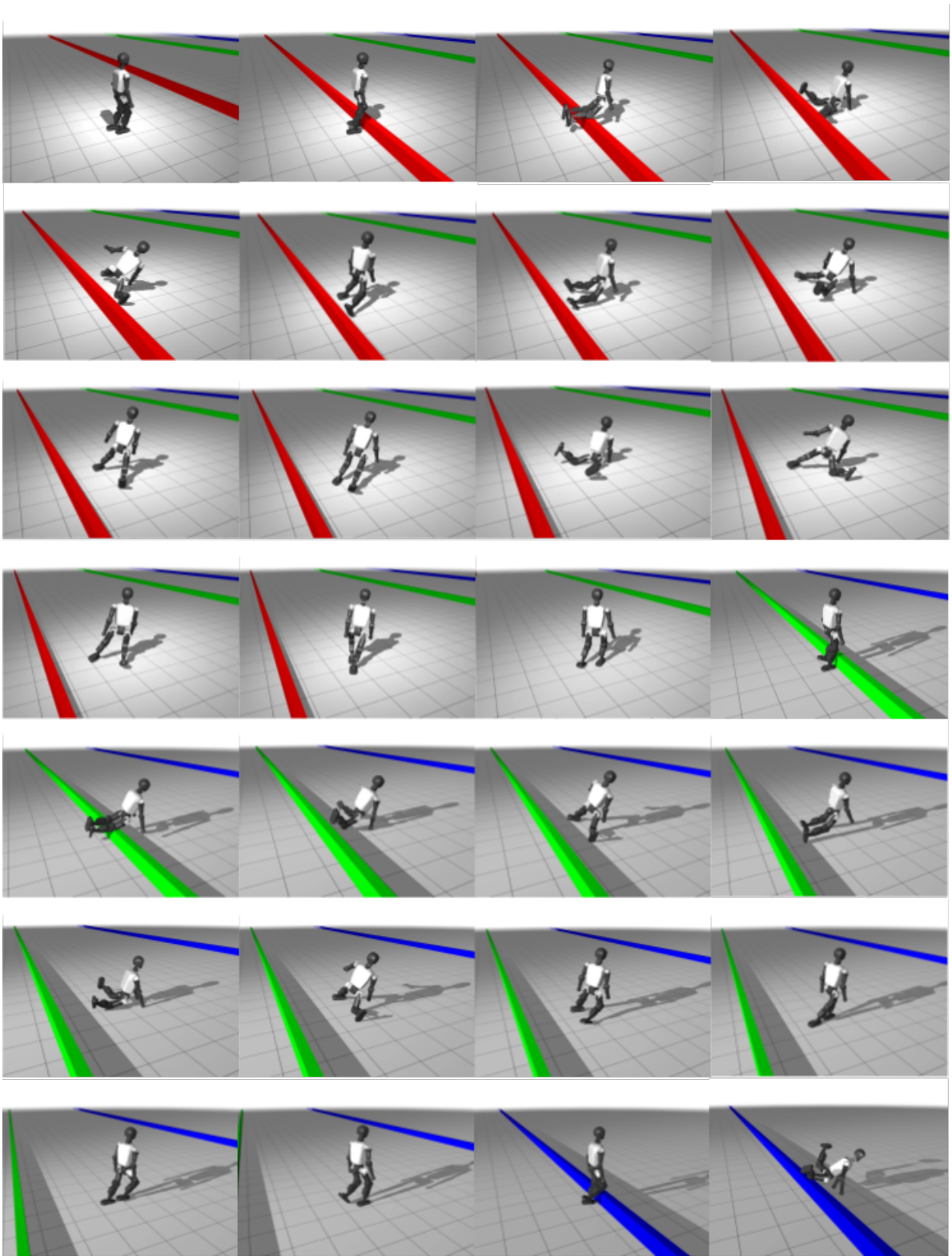

Figure A.16: (a) Baseline timed wall trial 4

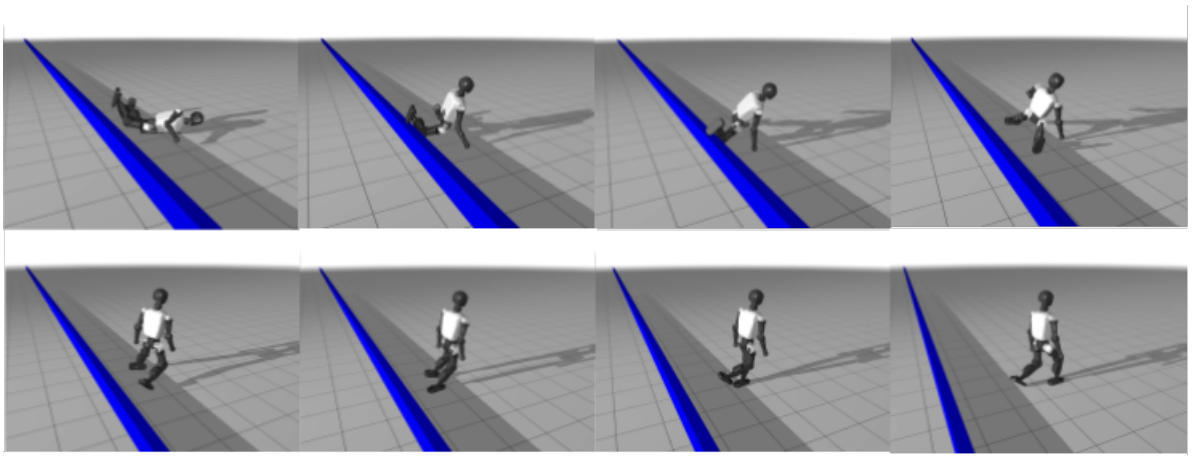

Figure A.17: (b) Baseline timed wall trial 4

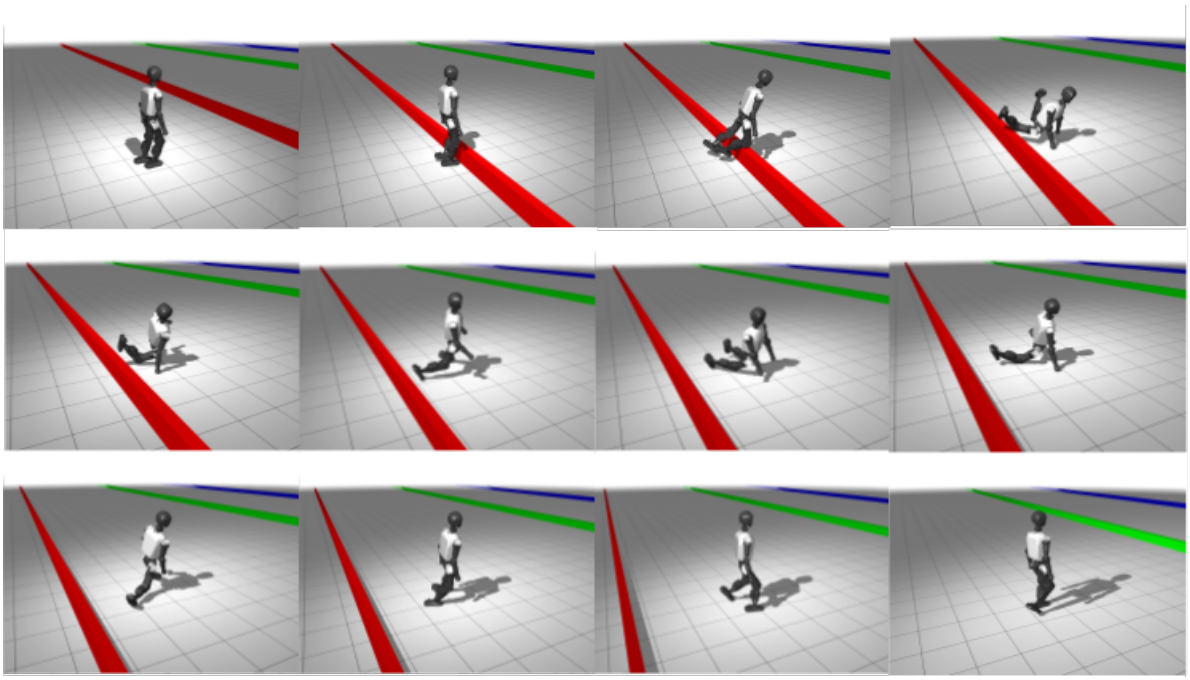

Figure A.18: (a) Baseline timed wall trial 5

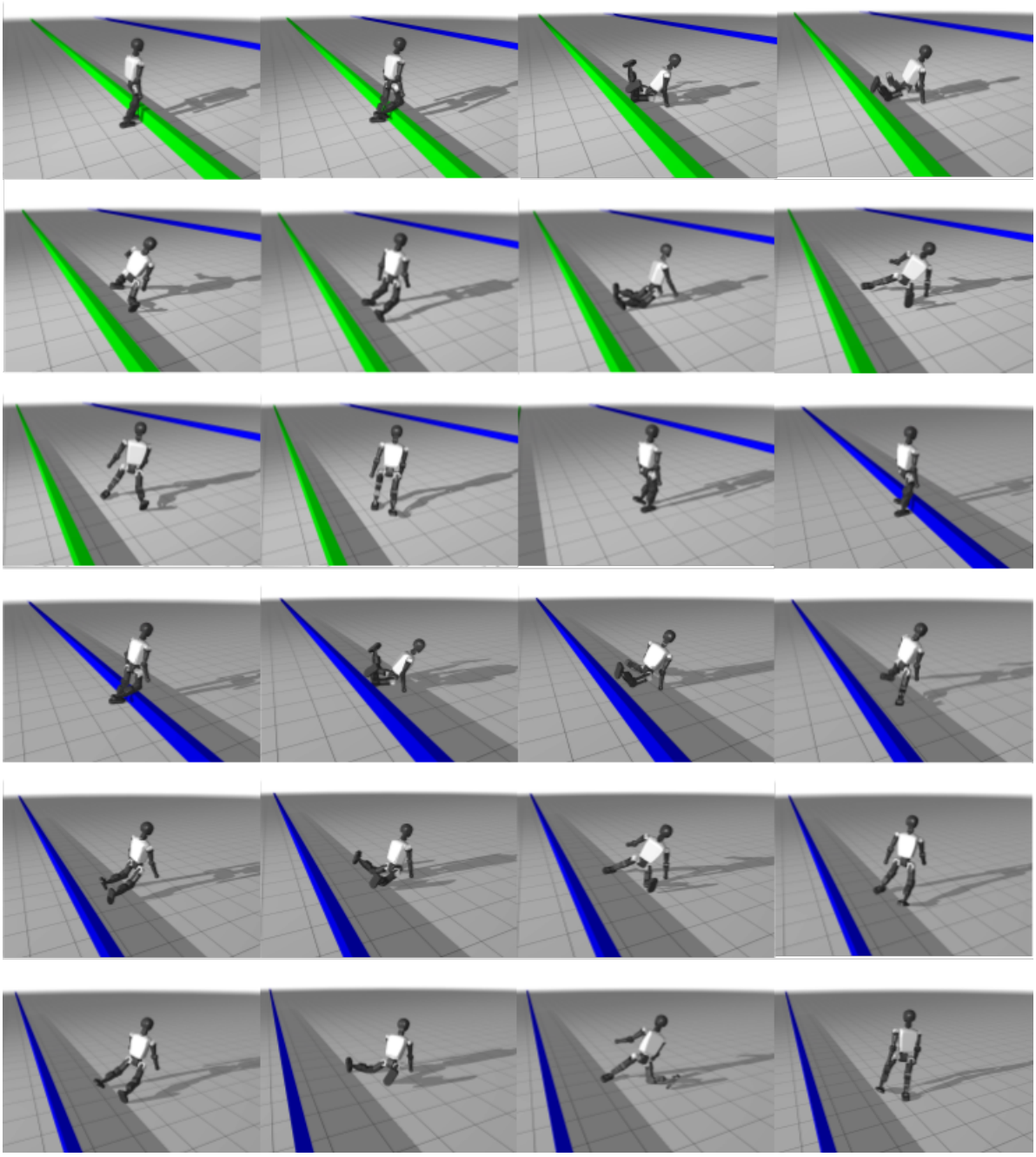

Figure A.19: (b) Baseline timed wall trial 5

Appendix B

Puddle Evaluation Trials

DAgger-100k Puddle Evaluations

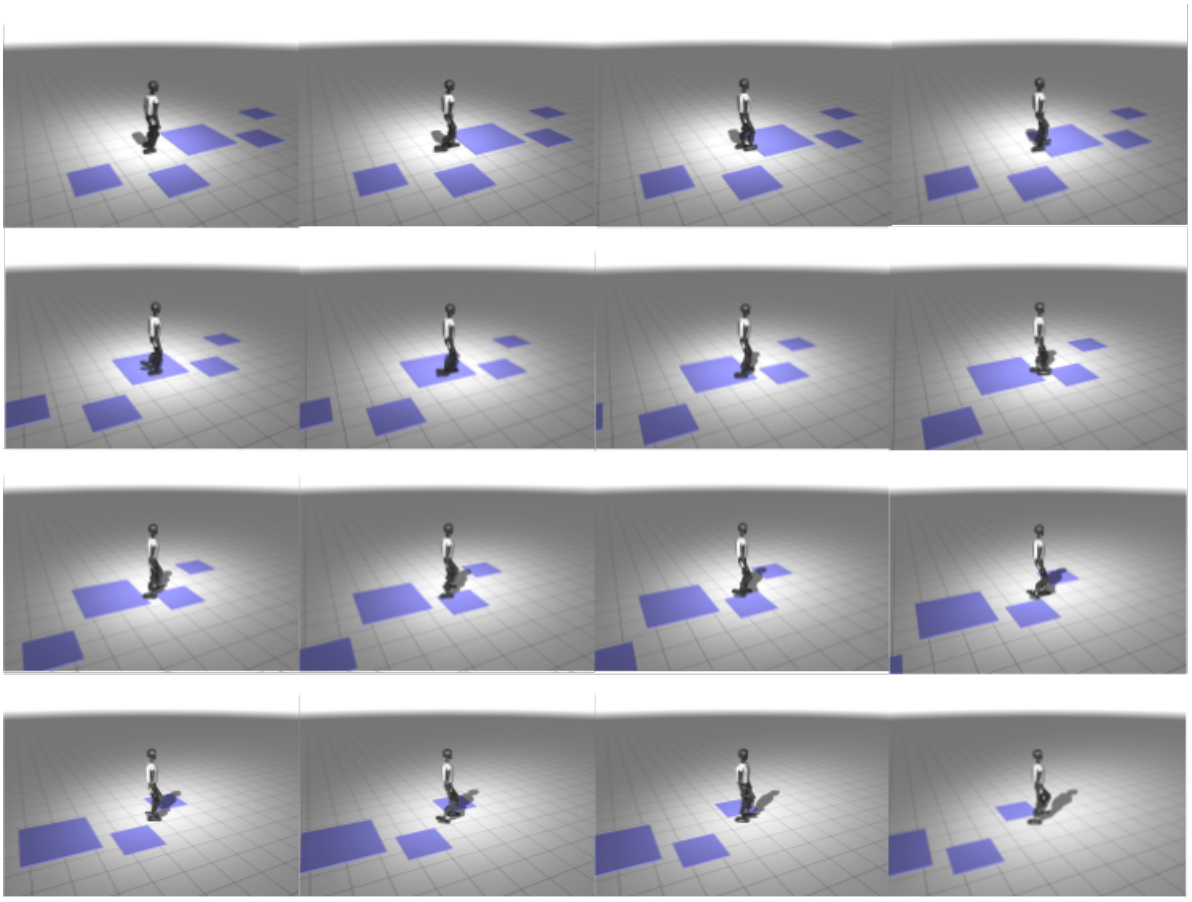

Figure B.1: DAgger-100k puddle evaluation 1

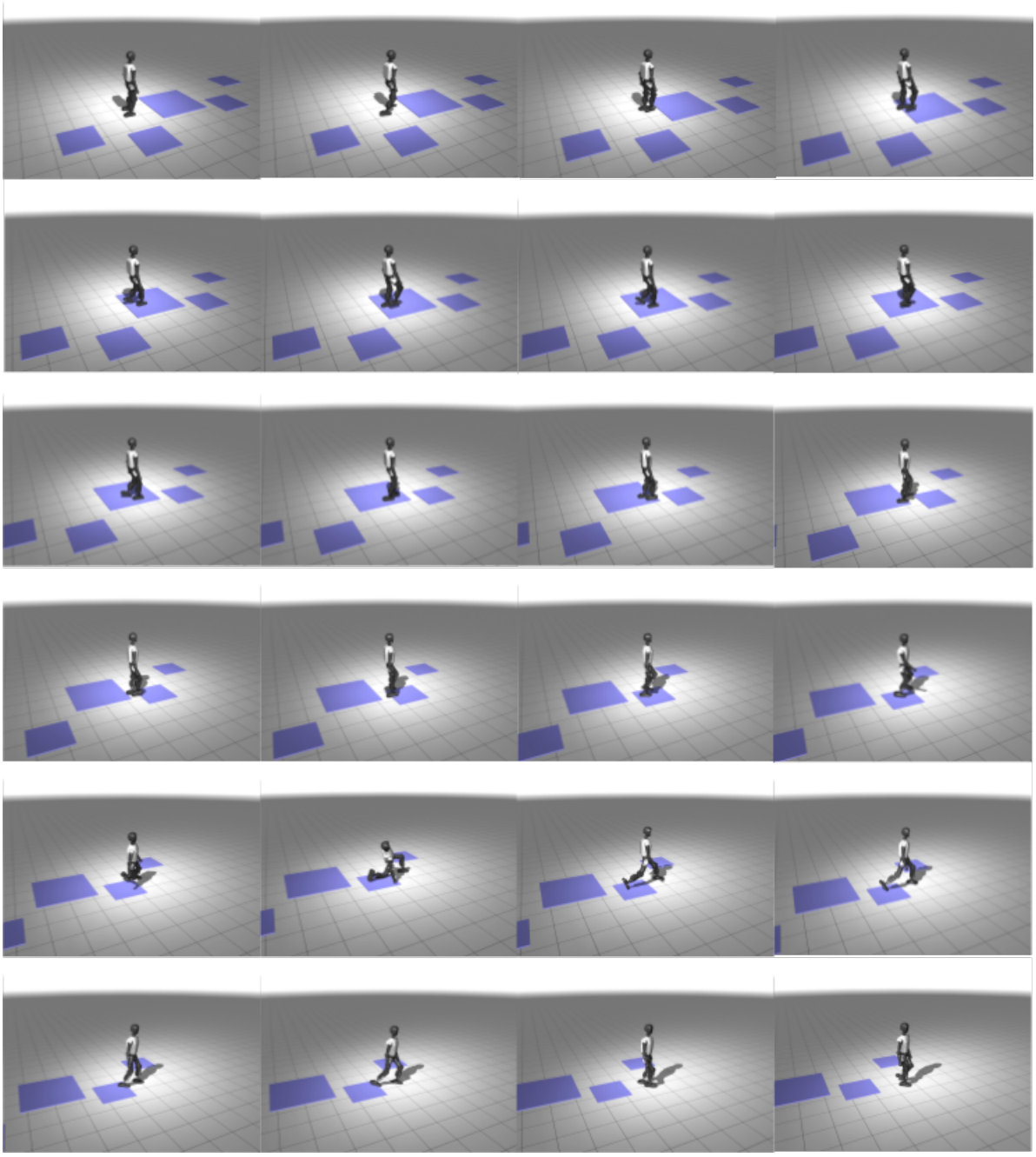

Figure B.2: DAgger-100k puddle evaluation 2

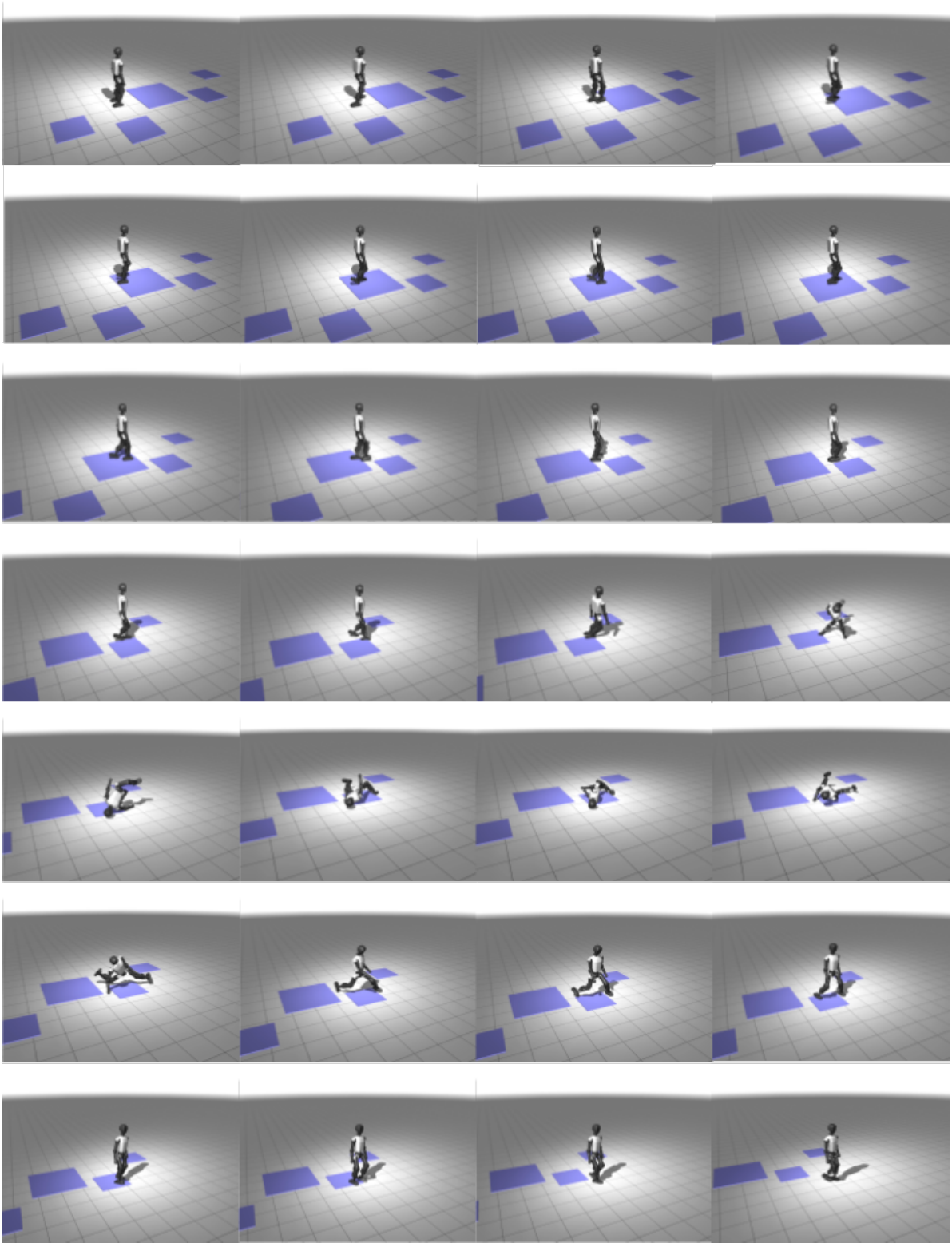

Figure B.3: DAgger-100k puddle evaluation 3

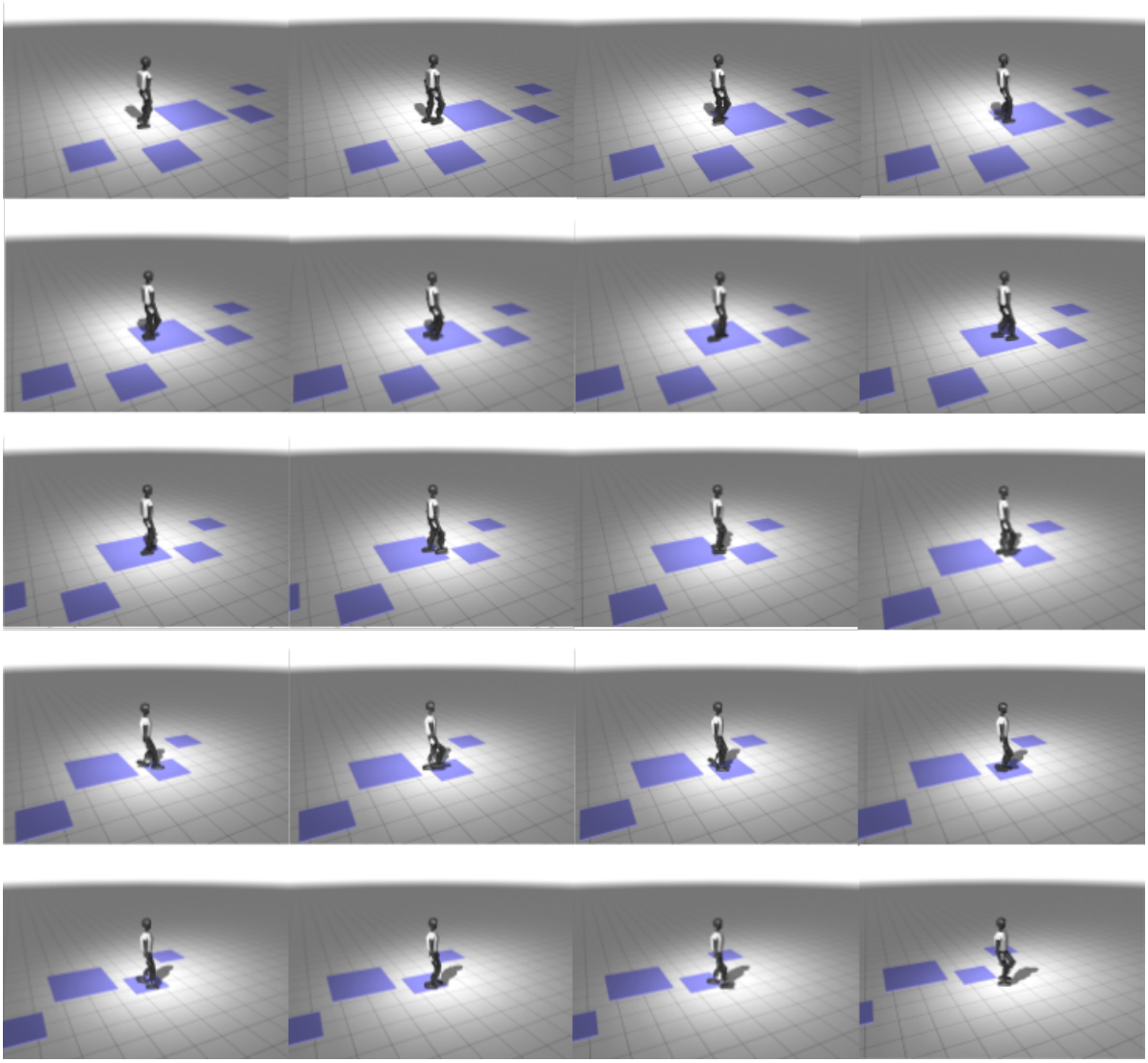

Figure B.4: DAgger-100k puddle evaluation 4

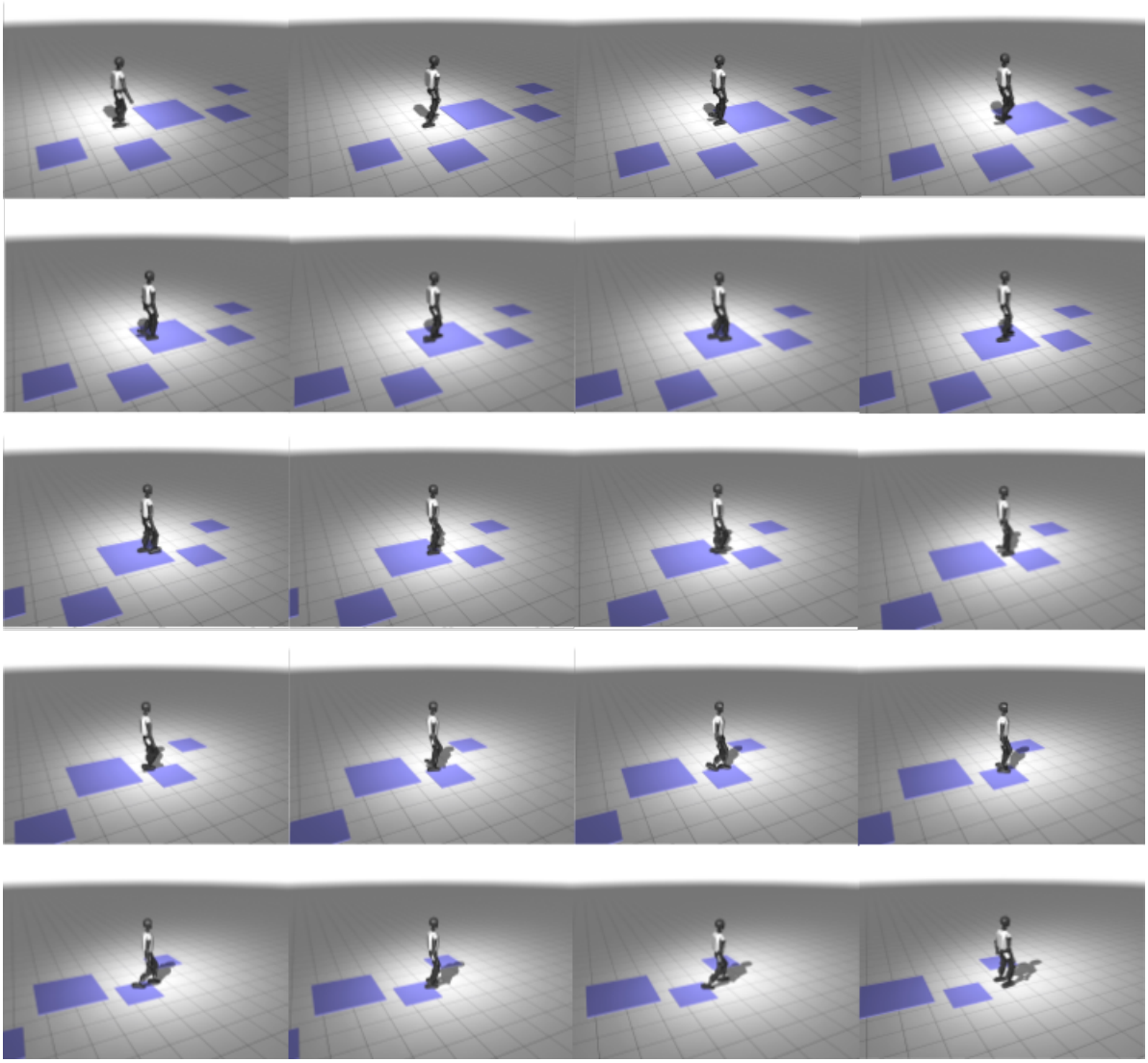

Figure B.5: DAgger-100k puddle evaluation 5

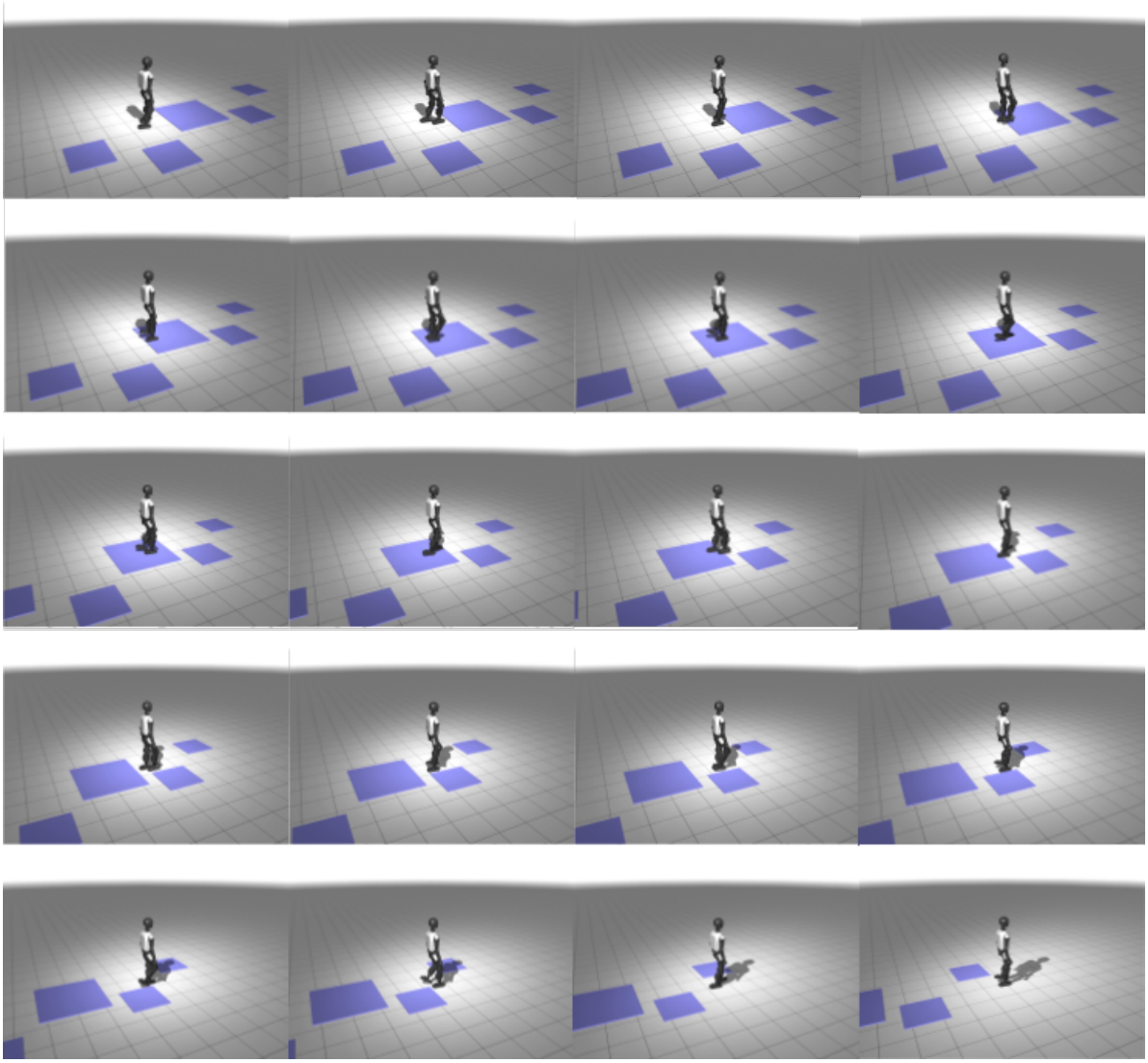

Figure B.6: DAgger-100k puddle evaluation 6

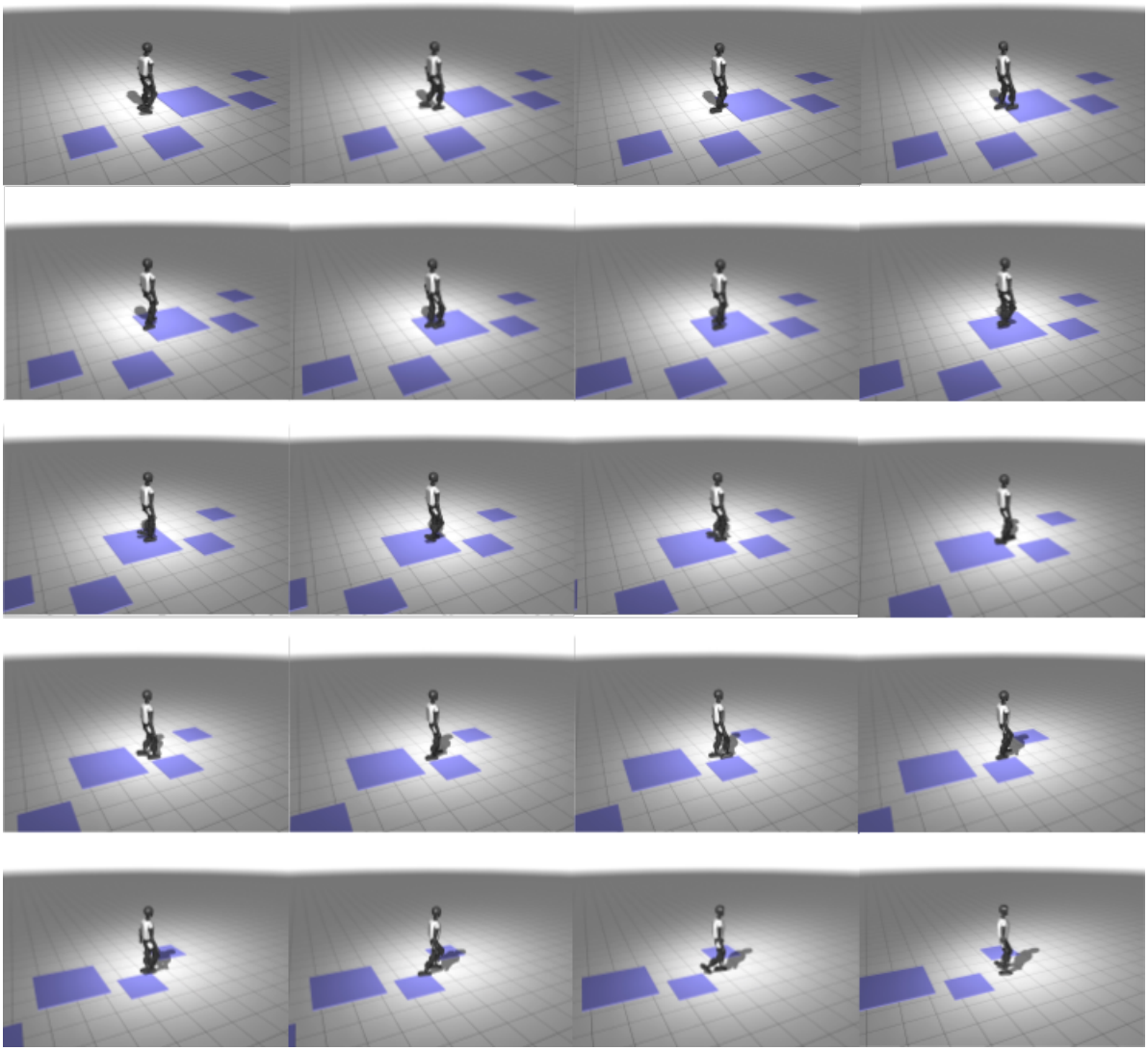

Figure B.7: DAgger-100k puddle evaluation 7

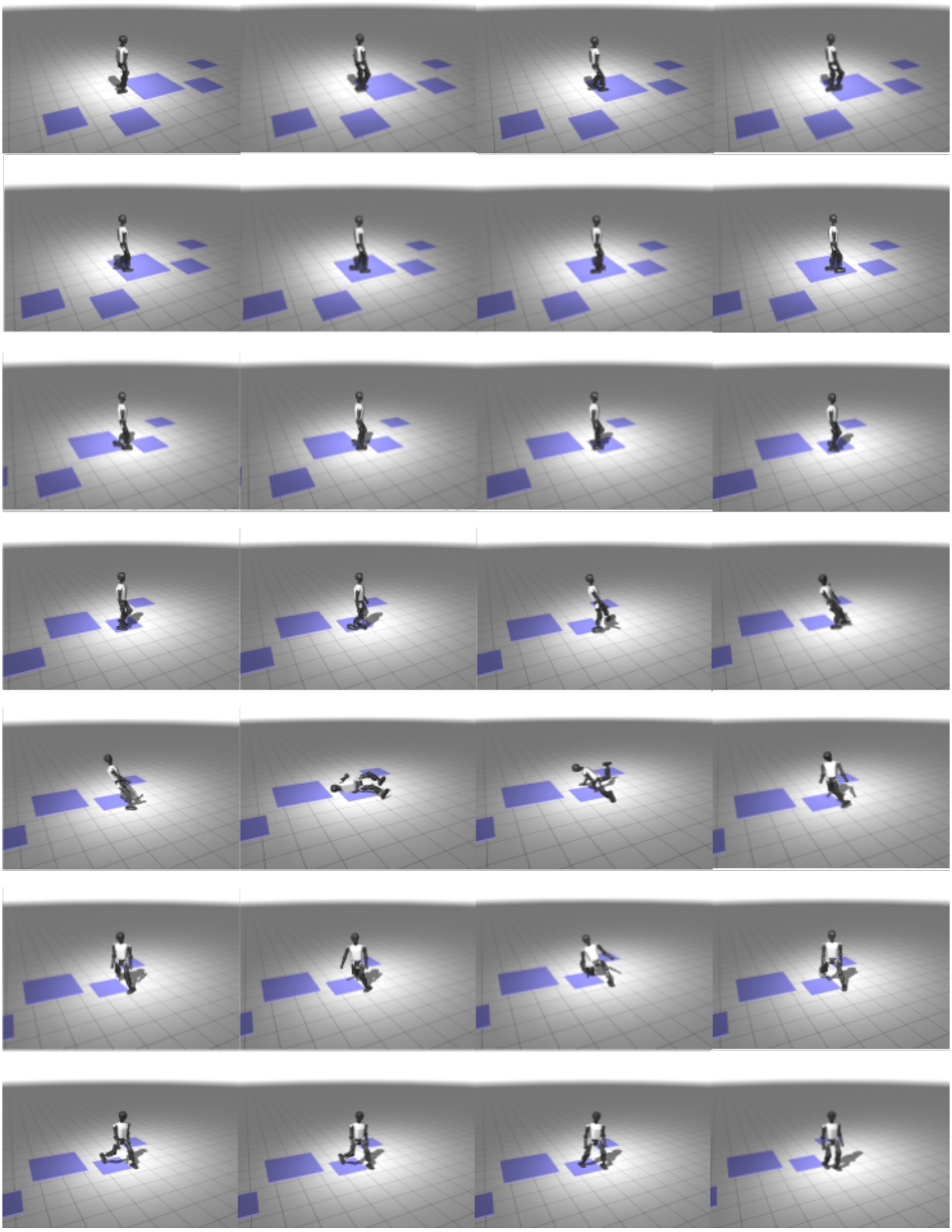

Figure B.8: DAgger-100k puddle evaluation 8

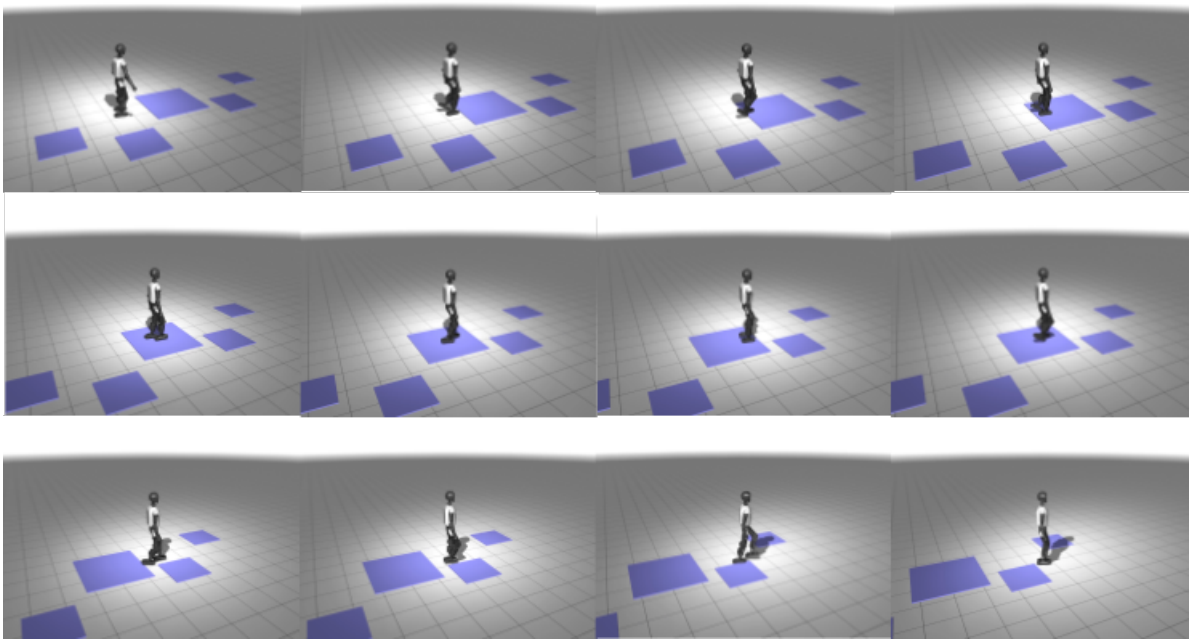

Figure B.9: DAgger-100k puddle evaluation 9

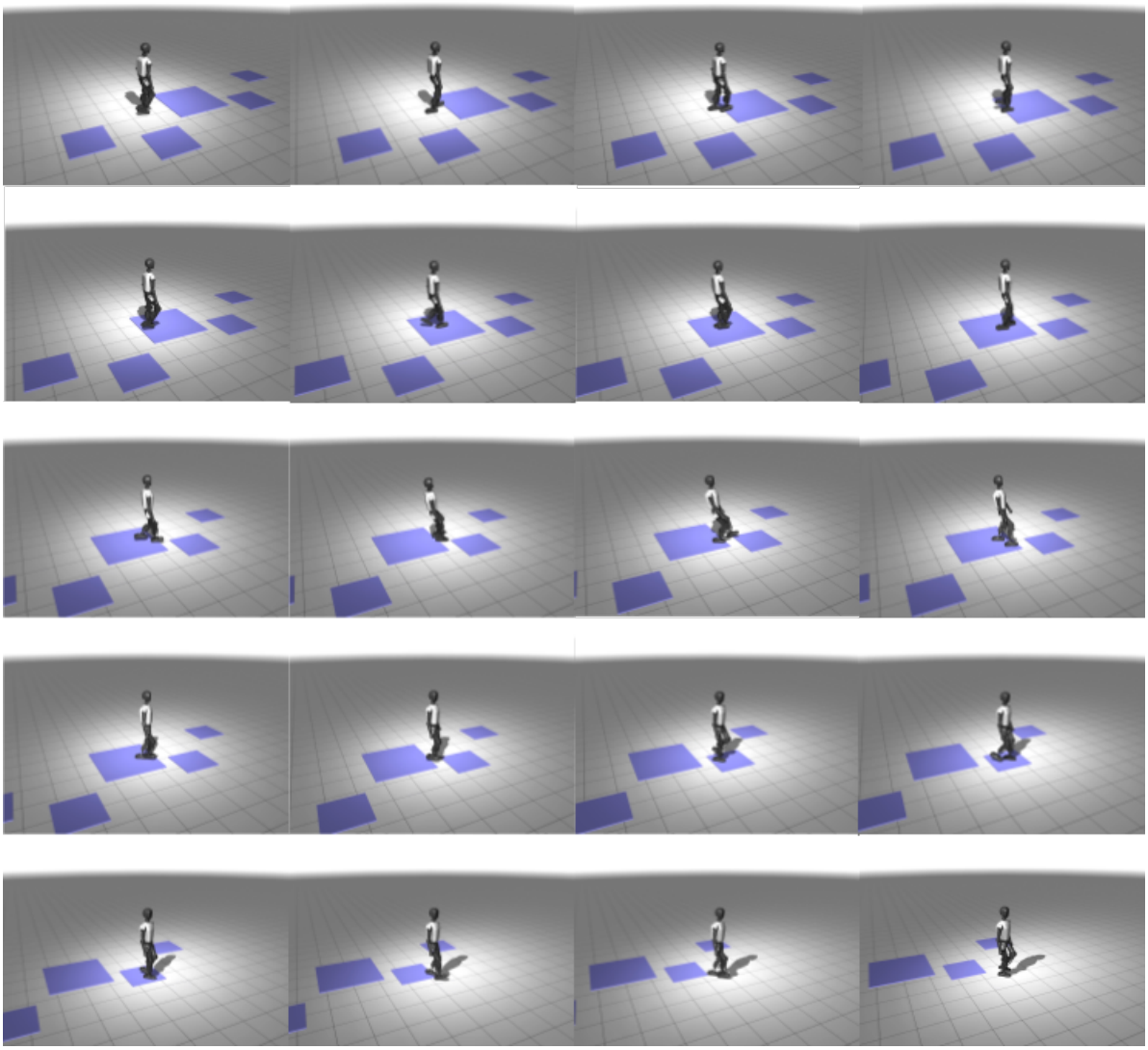

Figure B.10: DAgger-100k puddle evaluation 10

Baseline Puddle Evaluations

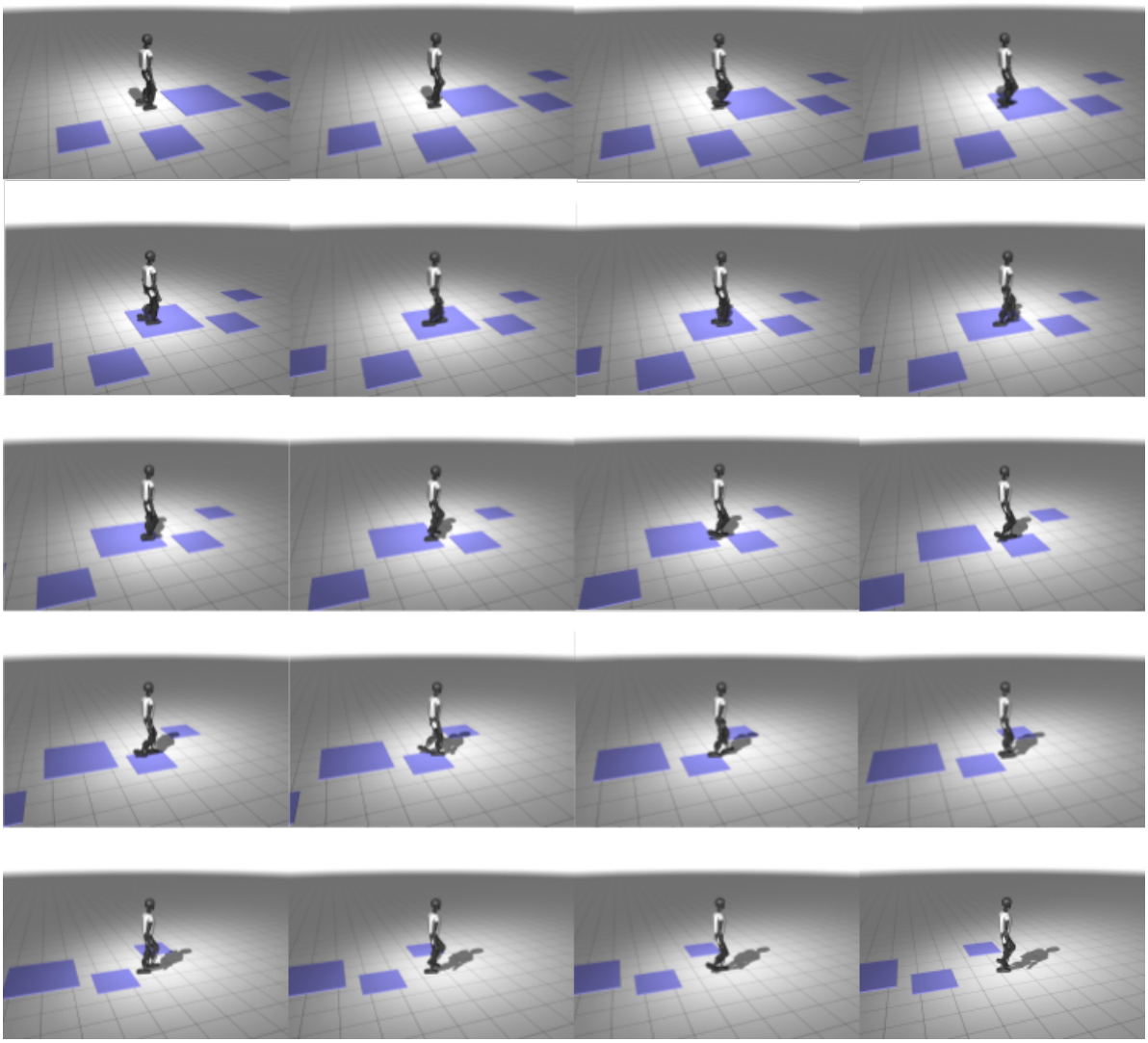

Figure B.11: Baseline puddle evaluation 1

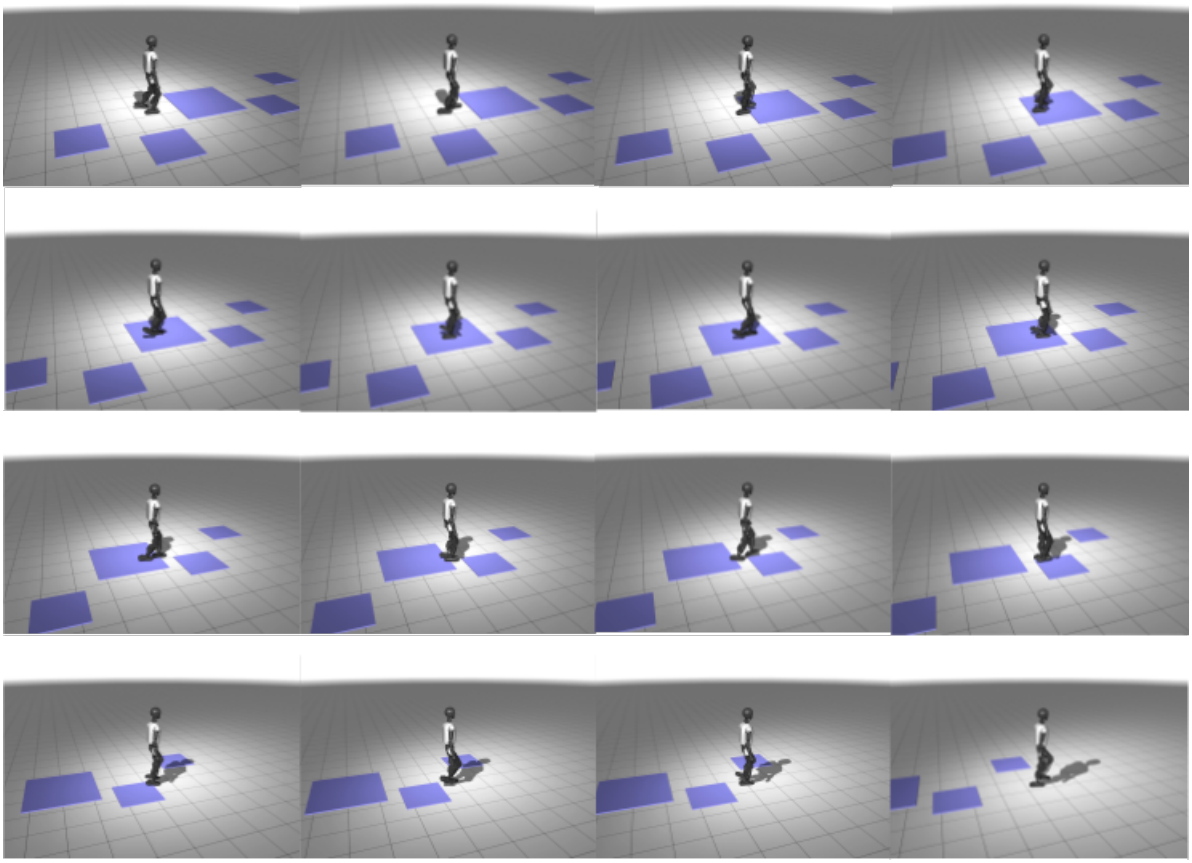

Figure B.12: Baseline puddle evaluation 2

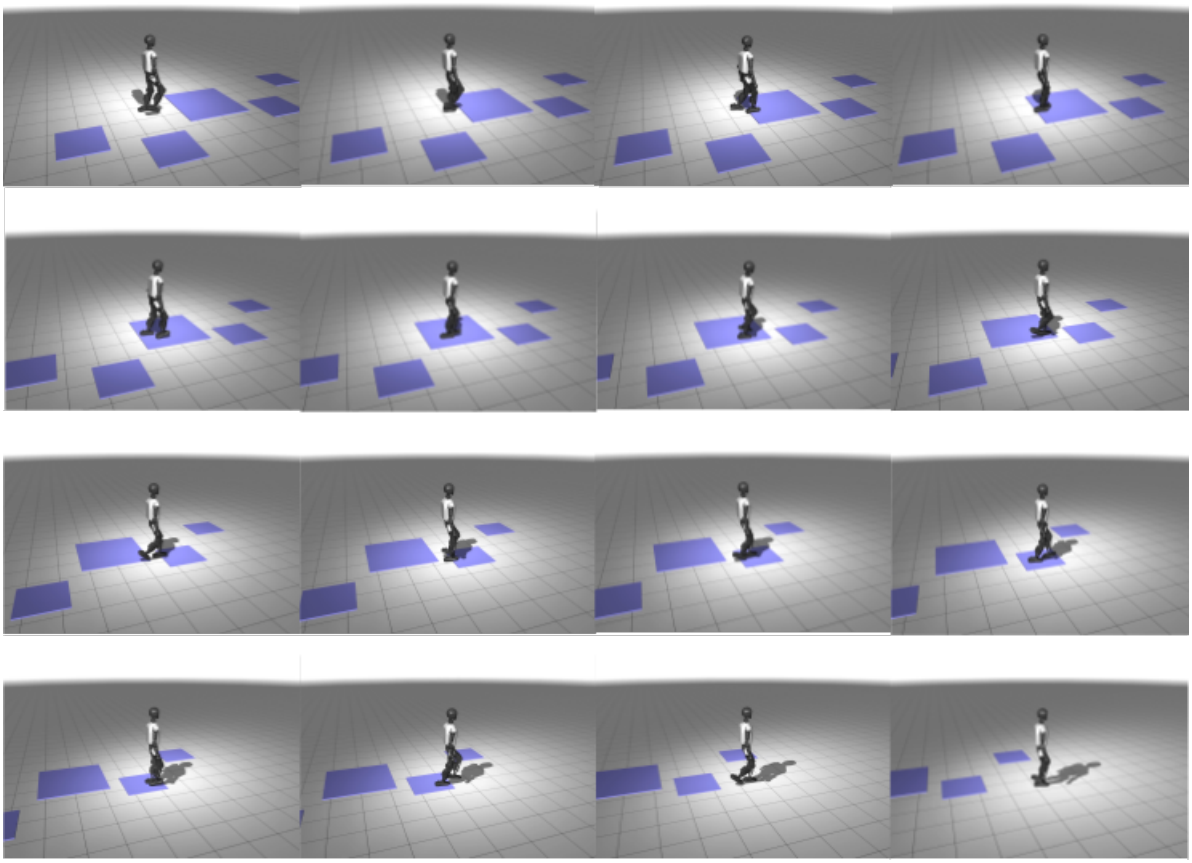

Figure B.13: Baseline puddle evaluation 3

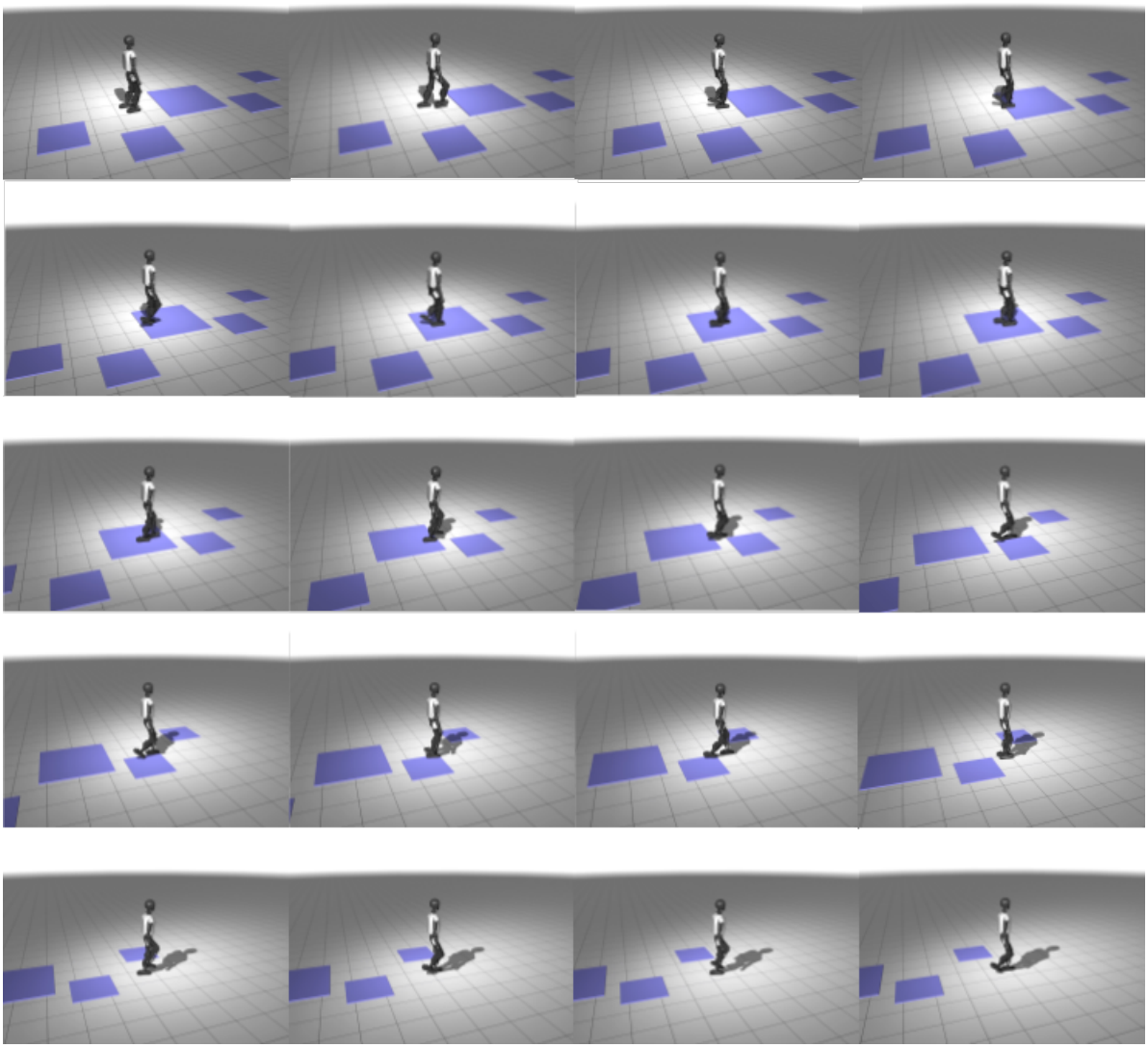

Figure B.14: Baseline puddle evaluation 4

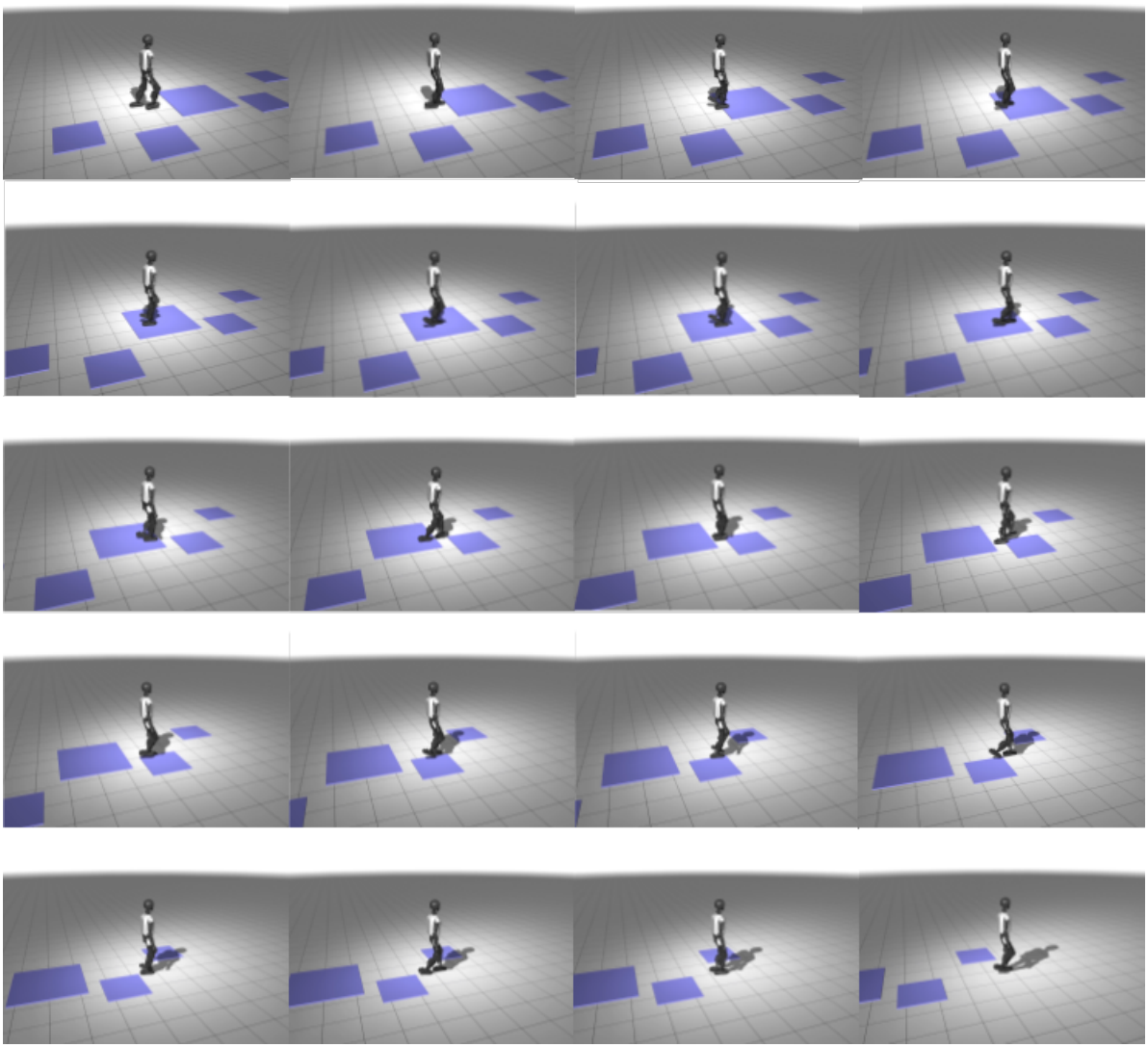

Figure B.15: Baseline puddle evaluation 5

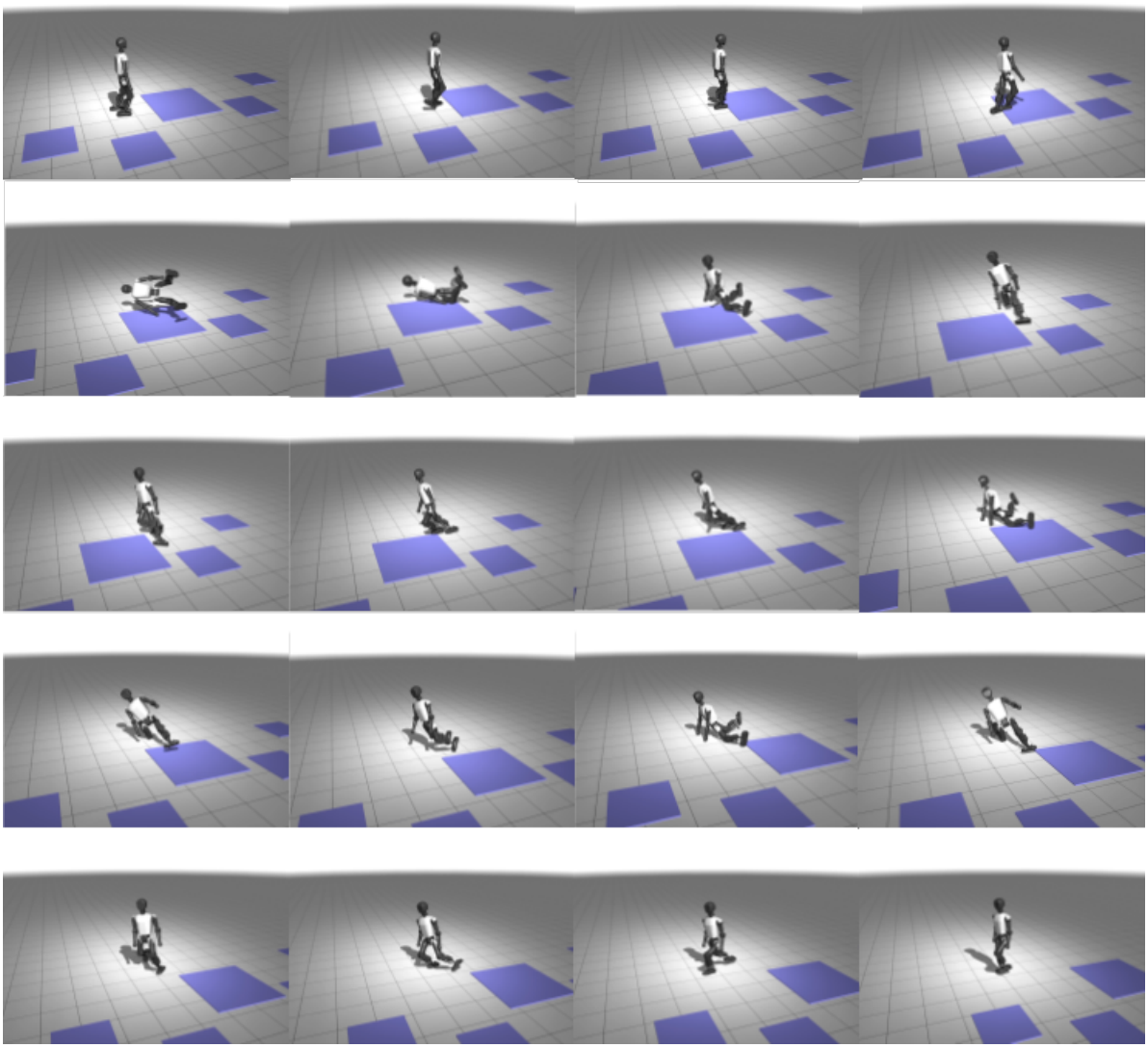

Figure B.16: Baseline puddle evaluation 6

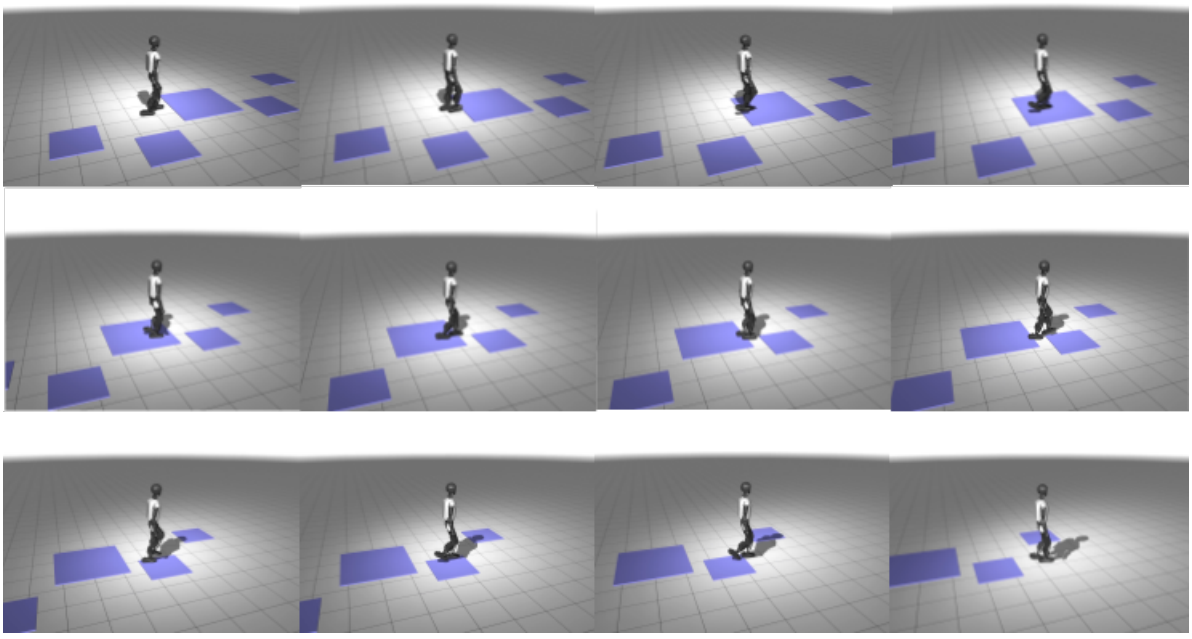

Figure B.17: Baseline puddle evaluation 7

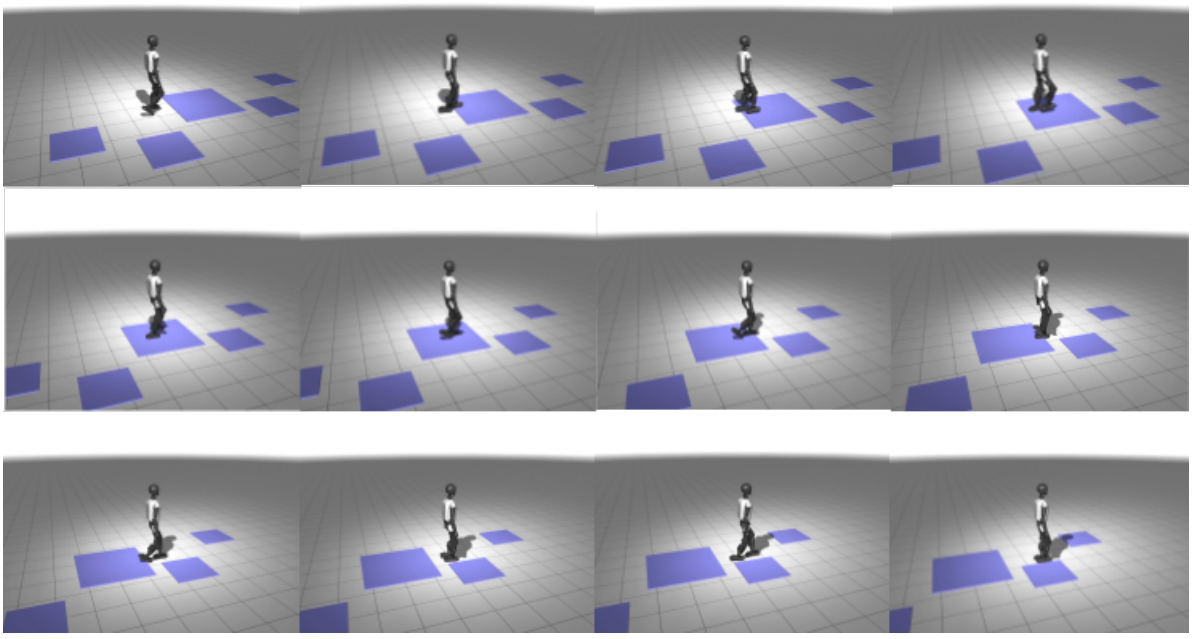

Figure B.18: Baseline puddle evaluation 8

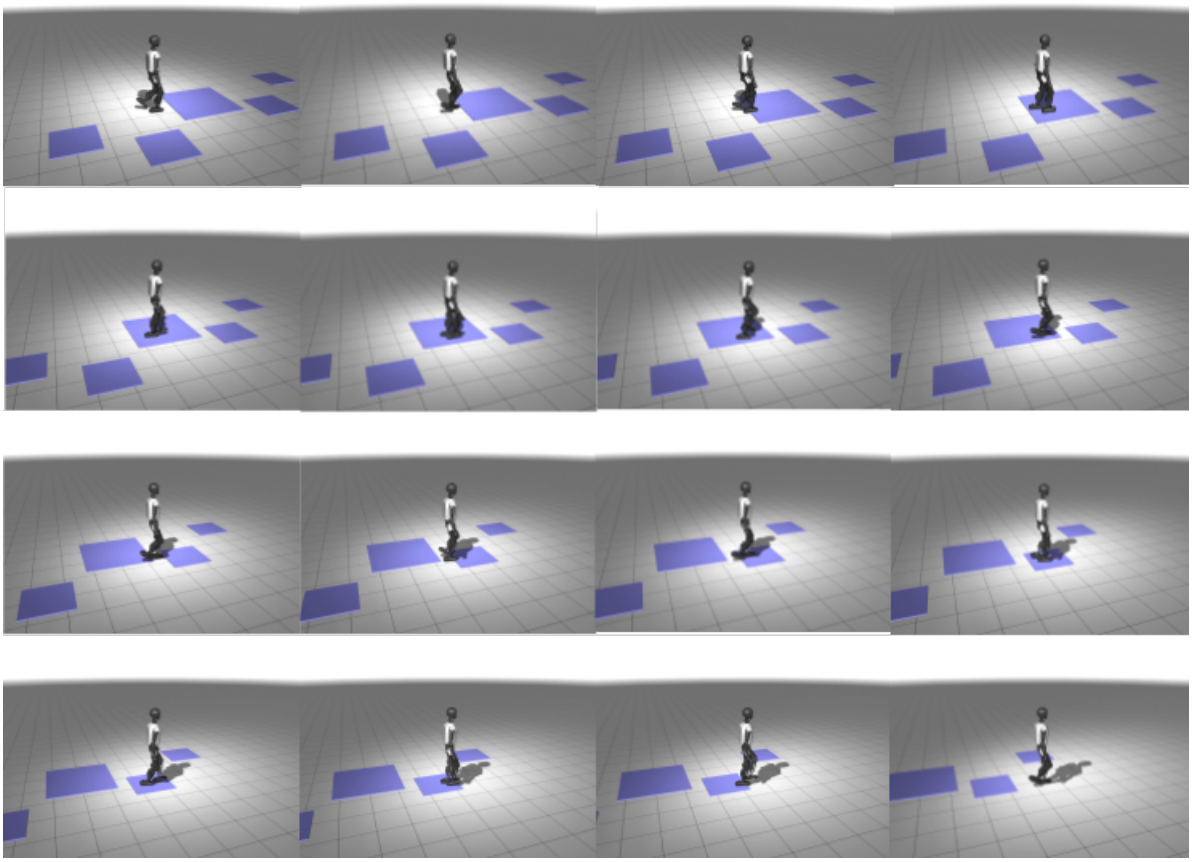

Figure B.19: Baseline puddle evaluation 9

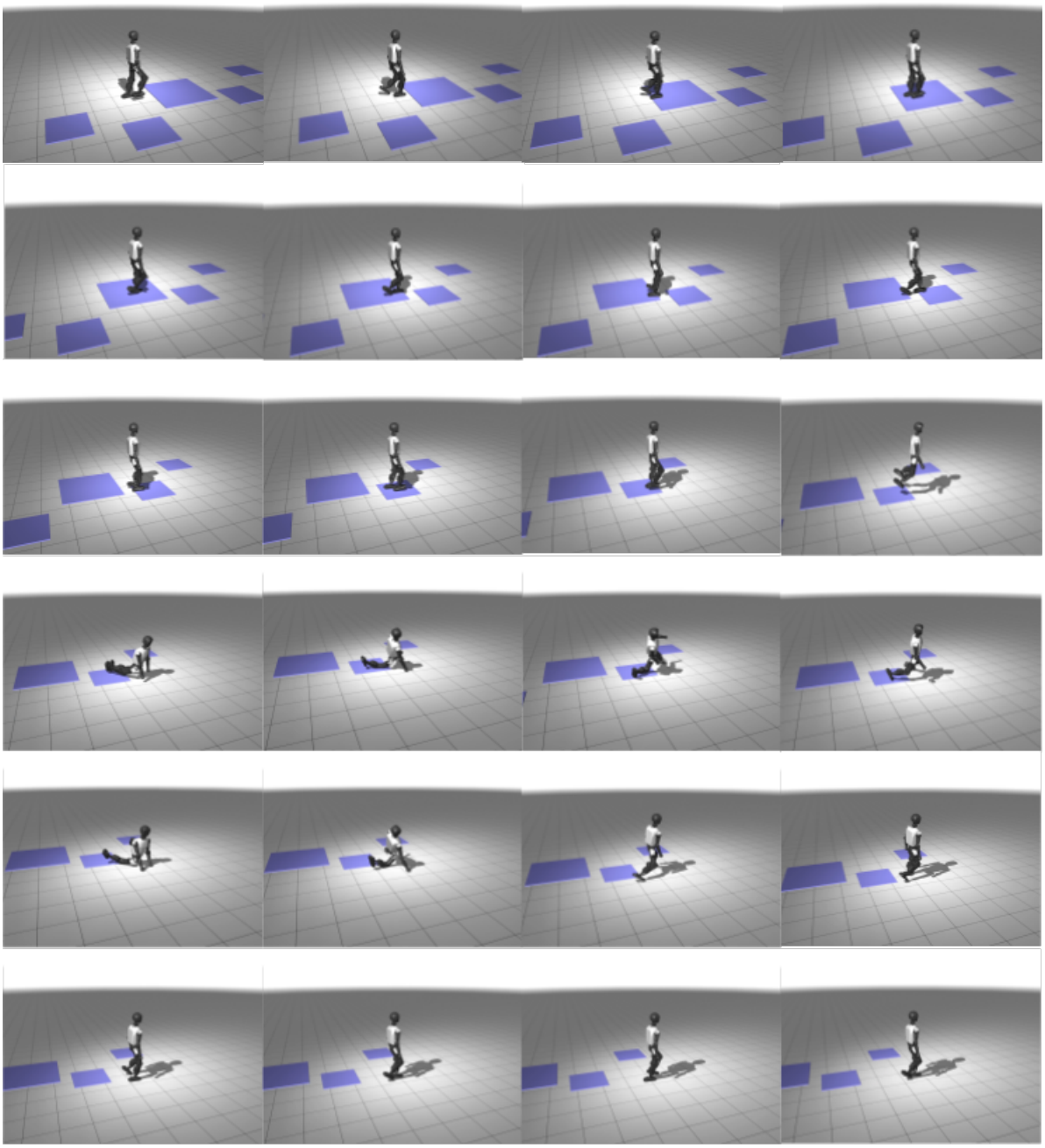

Figure B.20: Baseline puddle evaluation 10

Appendix C

Forward Push Evaluation Trials

DAgger-100k forward push evaluations

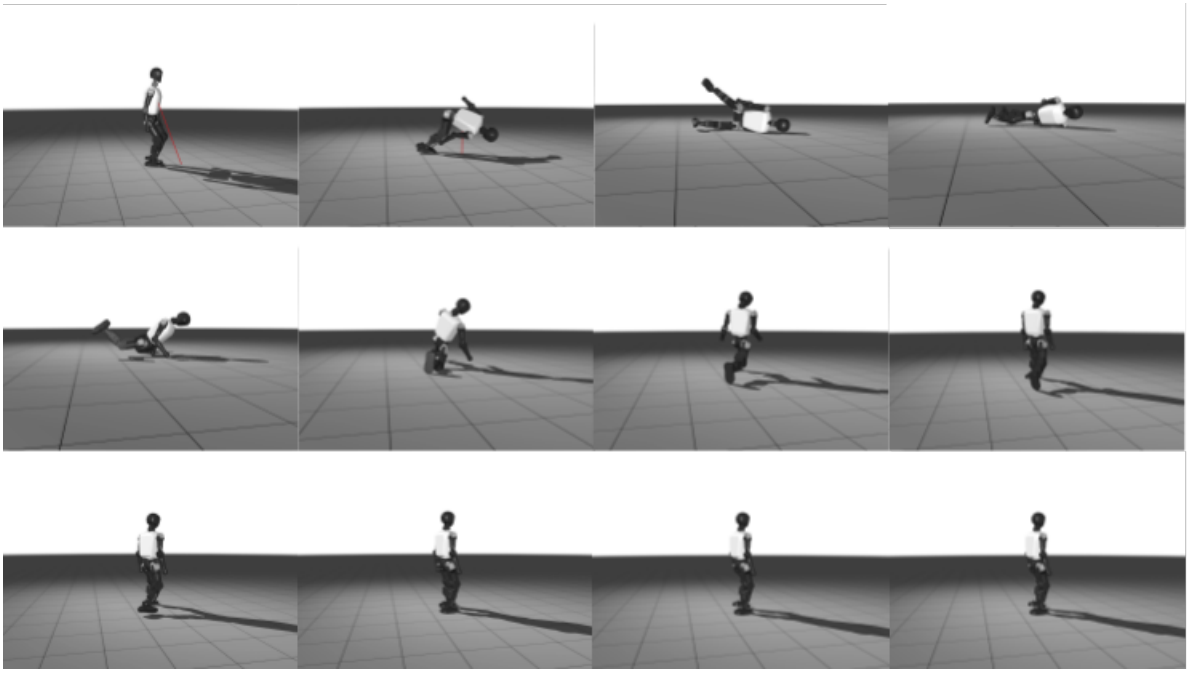

Figure C.1: DAgger-100k forward push evaluation 1

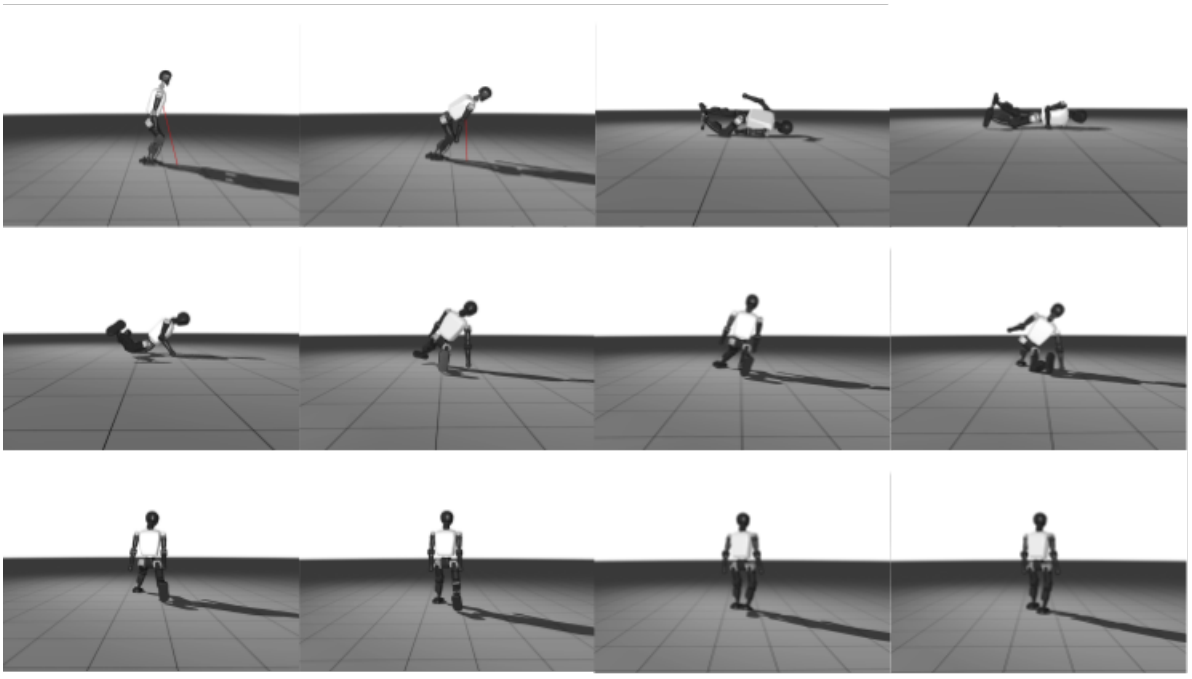

Figure C.2: DAgger-100k forward push evaluation 2

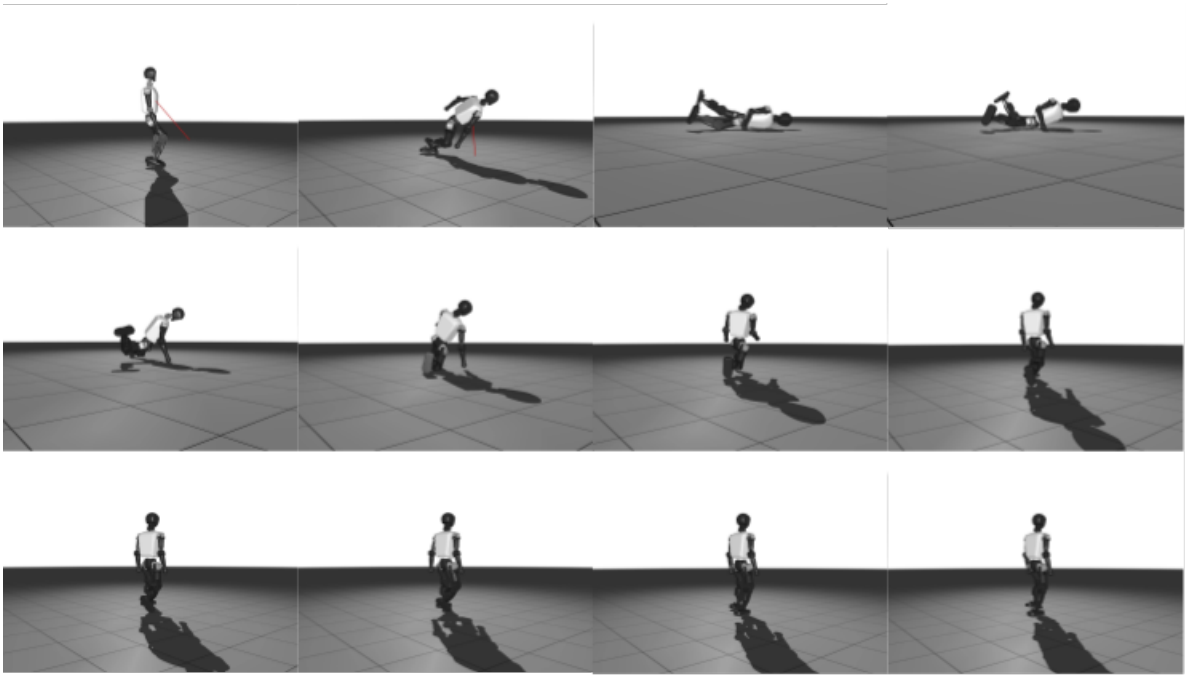

Figure C.3: DAgger-100k forward push evaluation 3

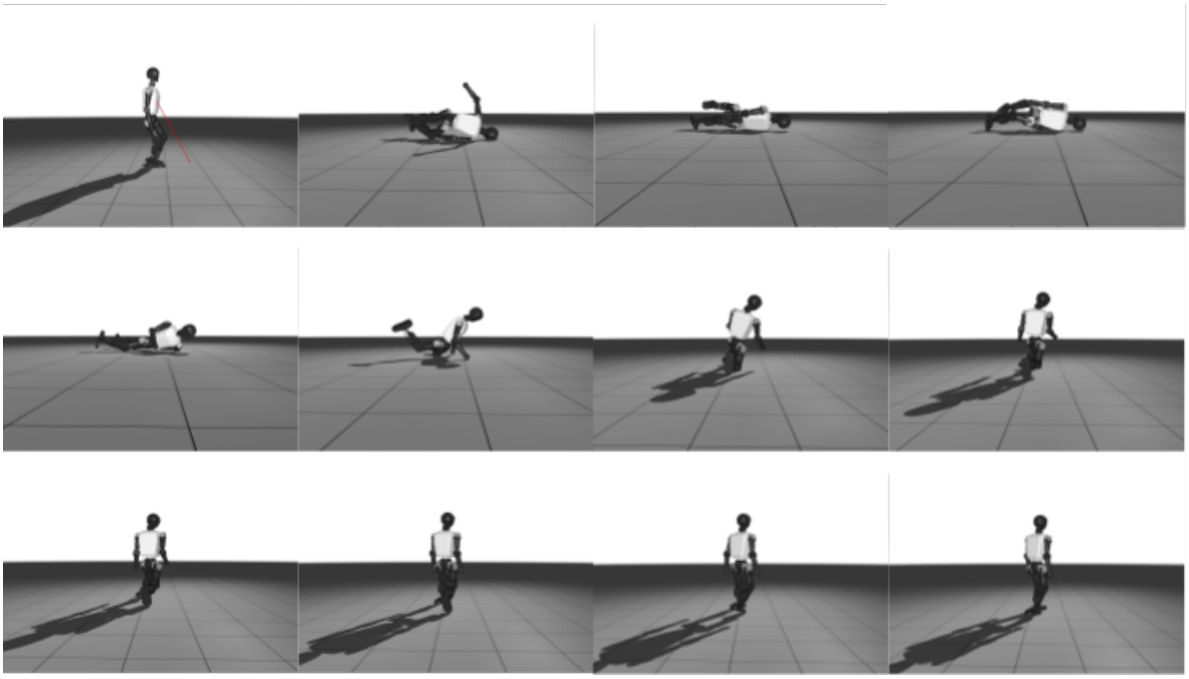

Figure C.4: DAgger-100k forward push evaluation 4

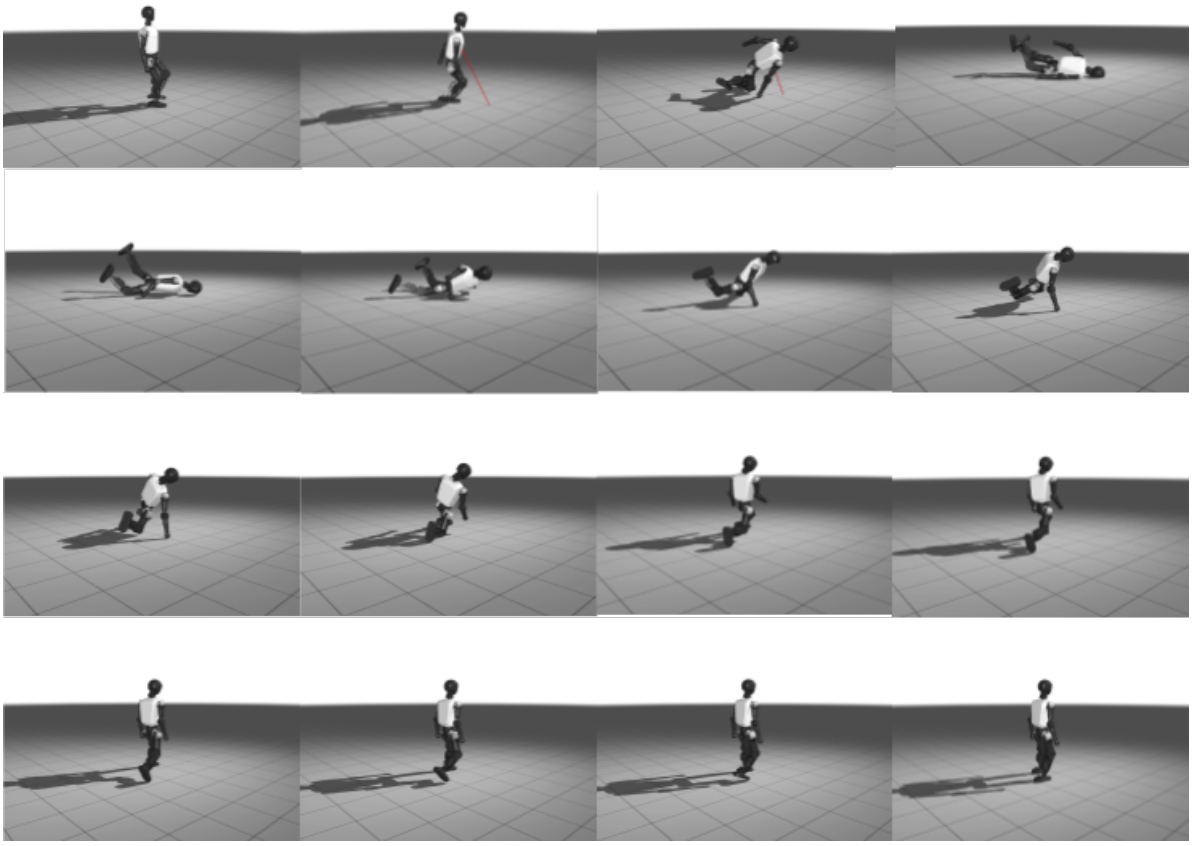

Figure C.5: DAgger-100k forward push evaluation 5

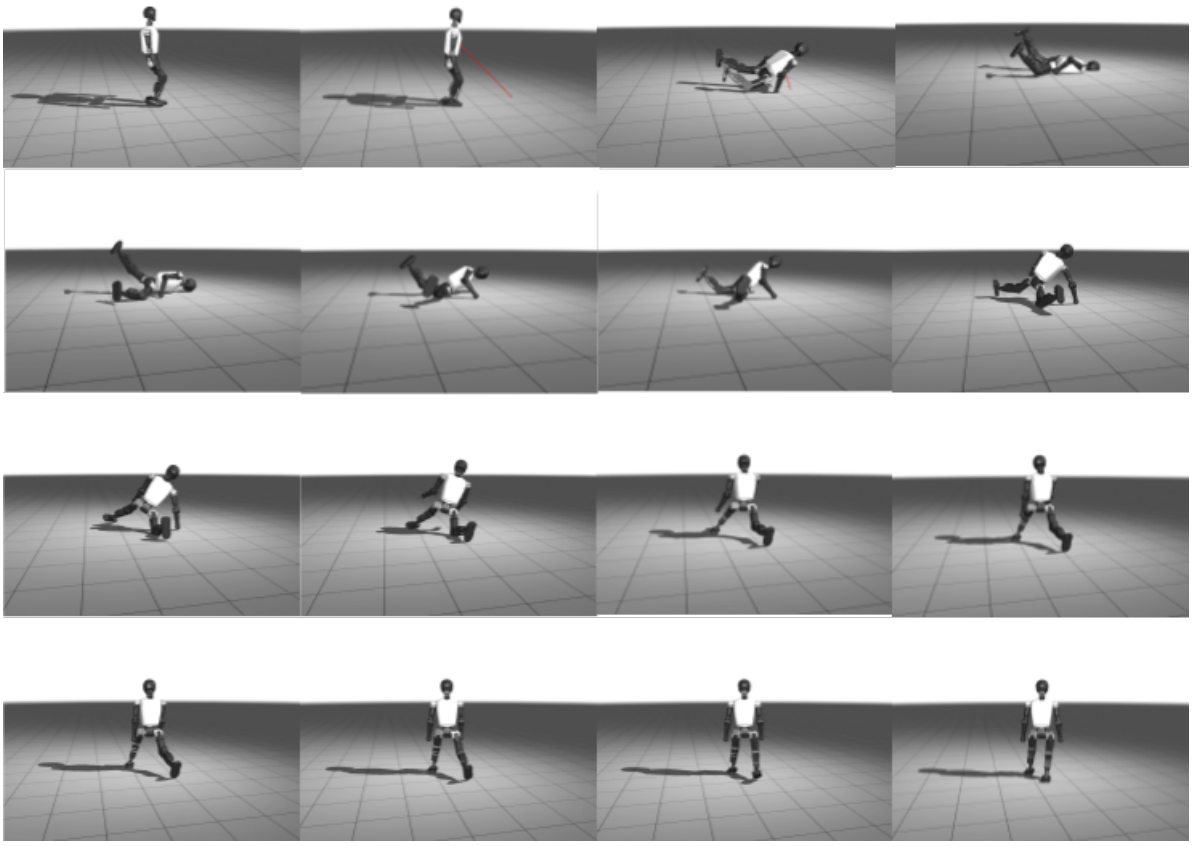

Figure C.6: DAgger-100k forward push evaluation 6

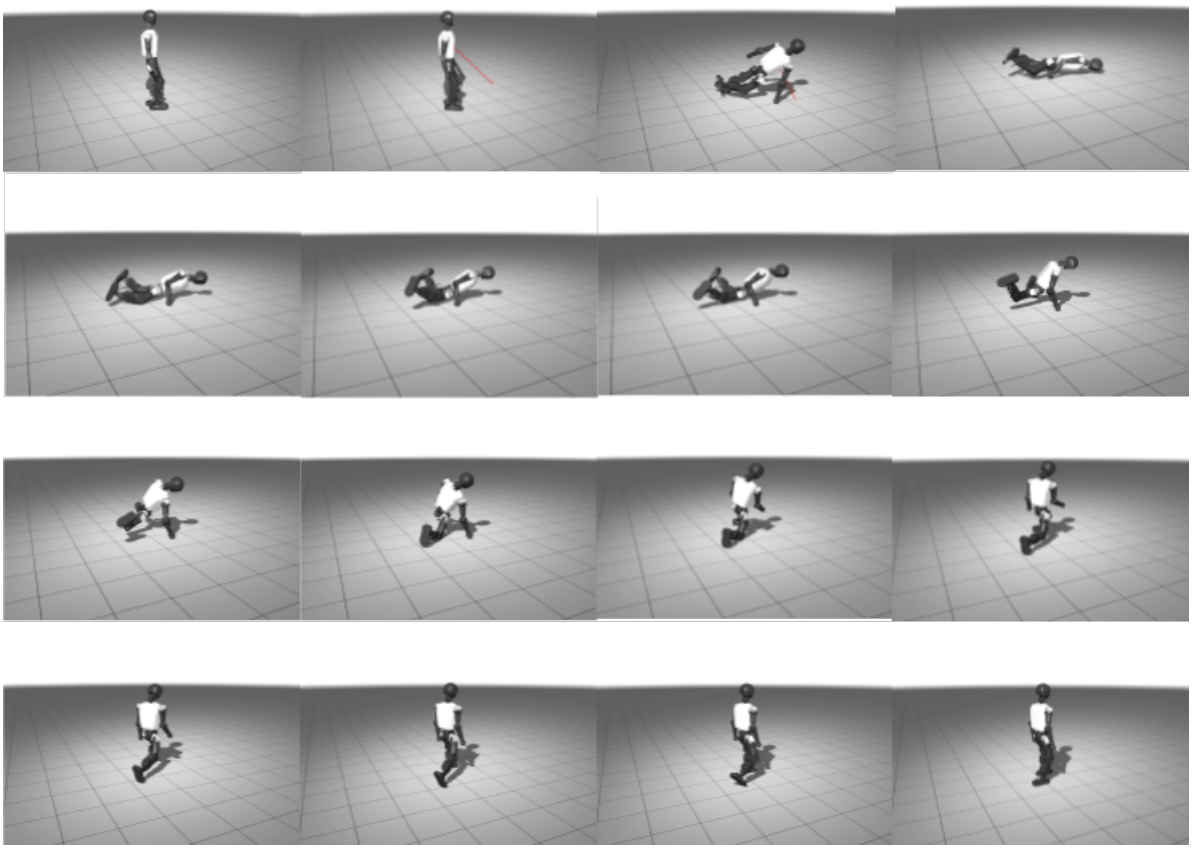

Figure C.7: DAgger-100k forward push evaluation 7

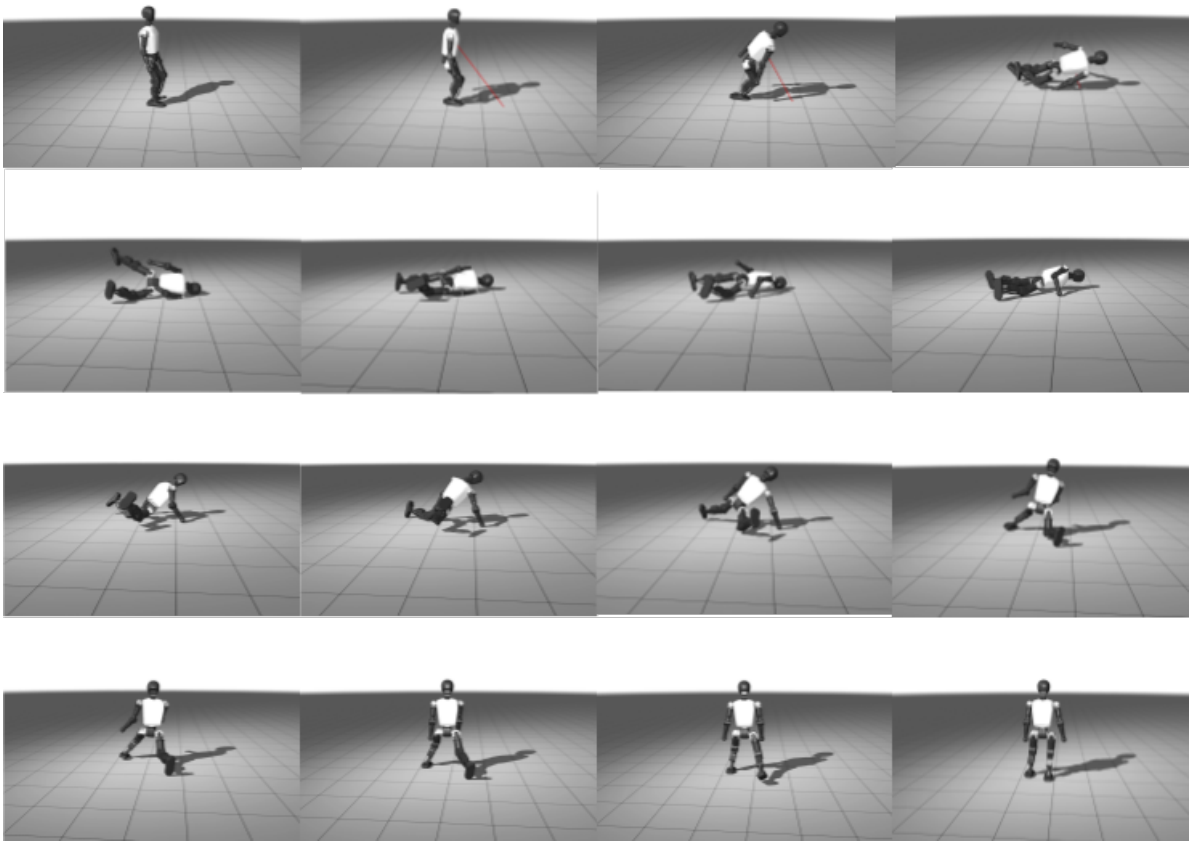

Figure C.8: DAgger-100k forward push evaluation 8

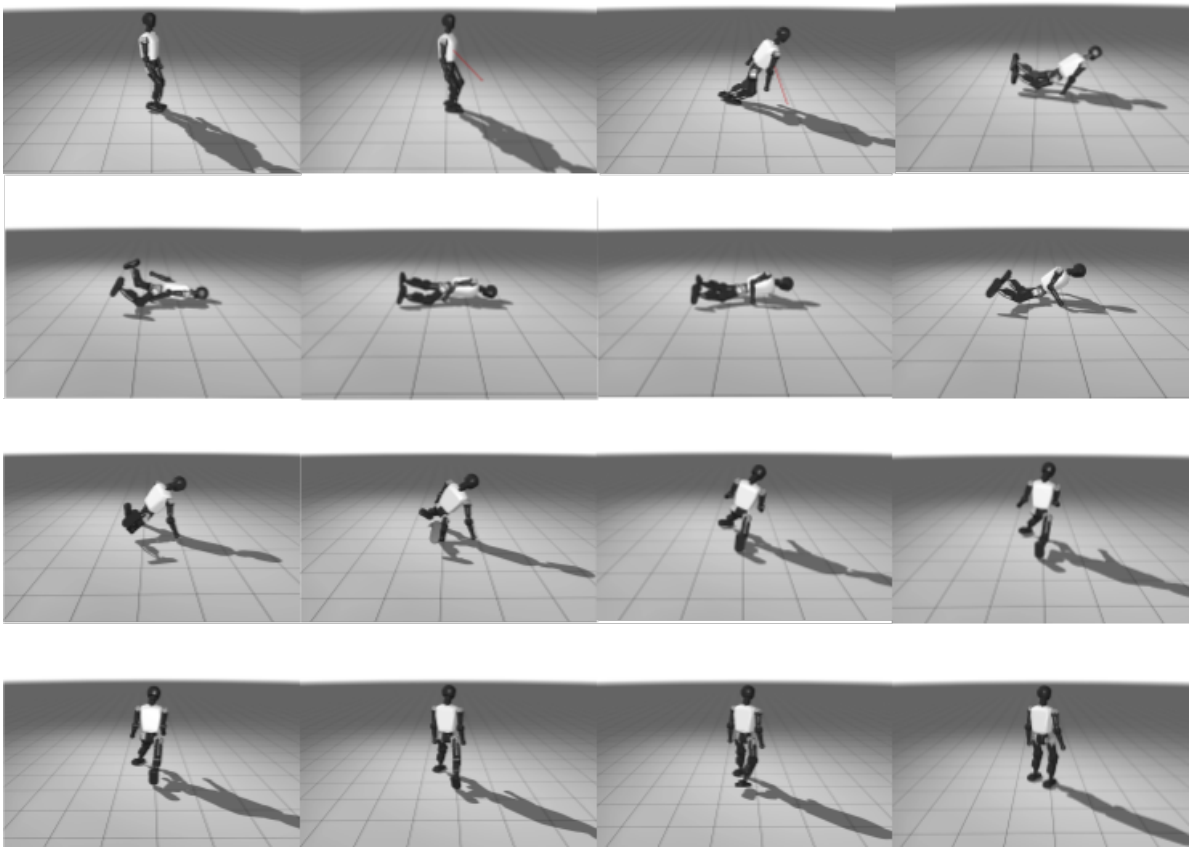

Figure C.9: DAgger-100k forward push evaluation 9

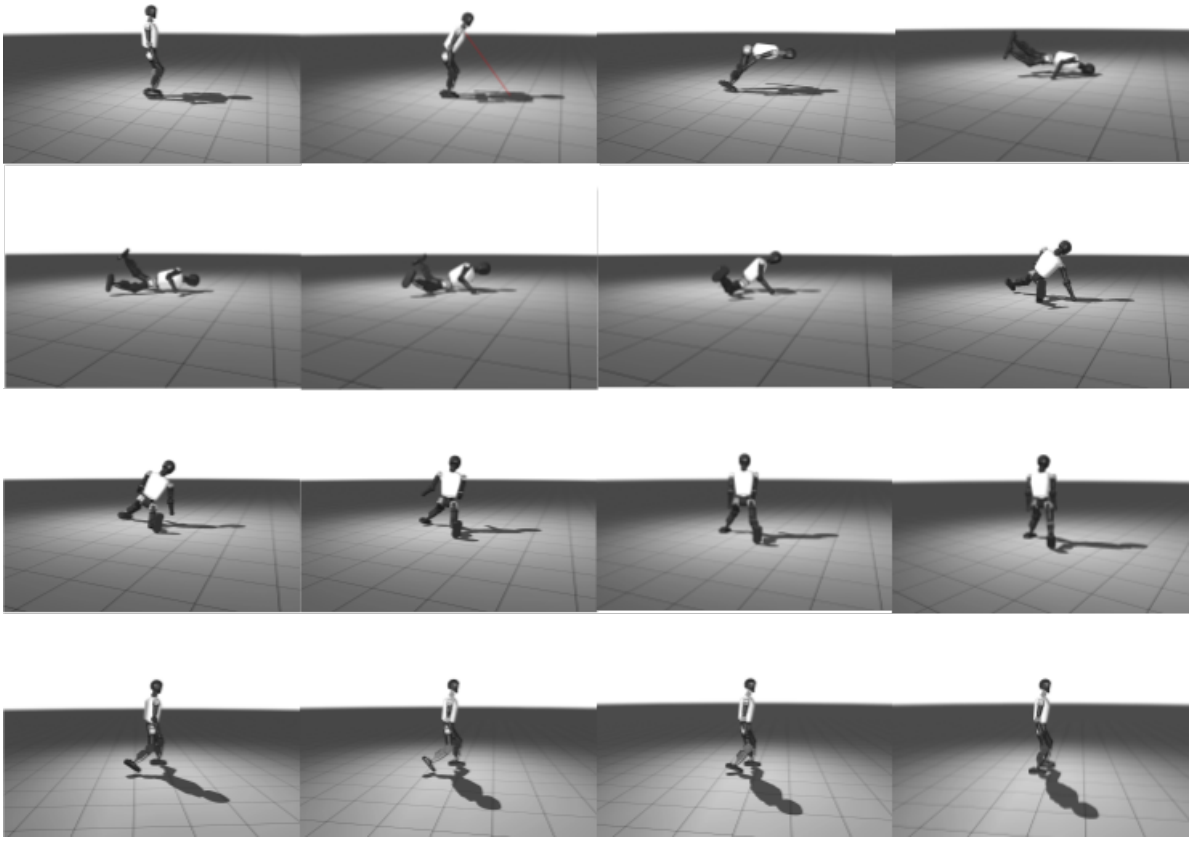

Figure C.10: DAgger-100k forward push evaluation 10

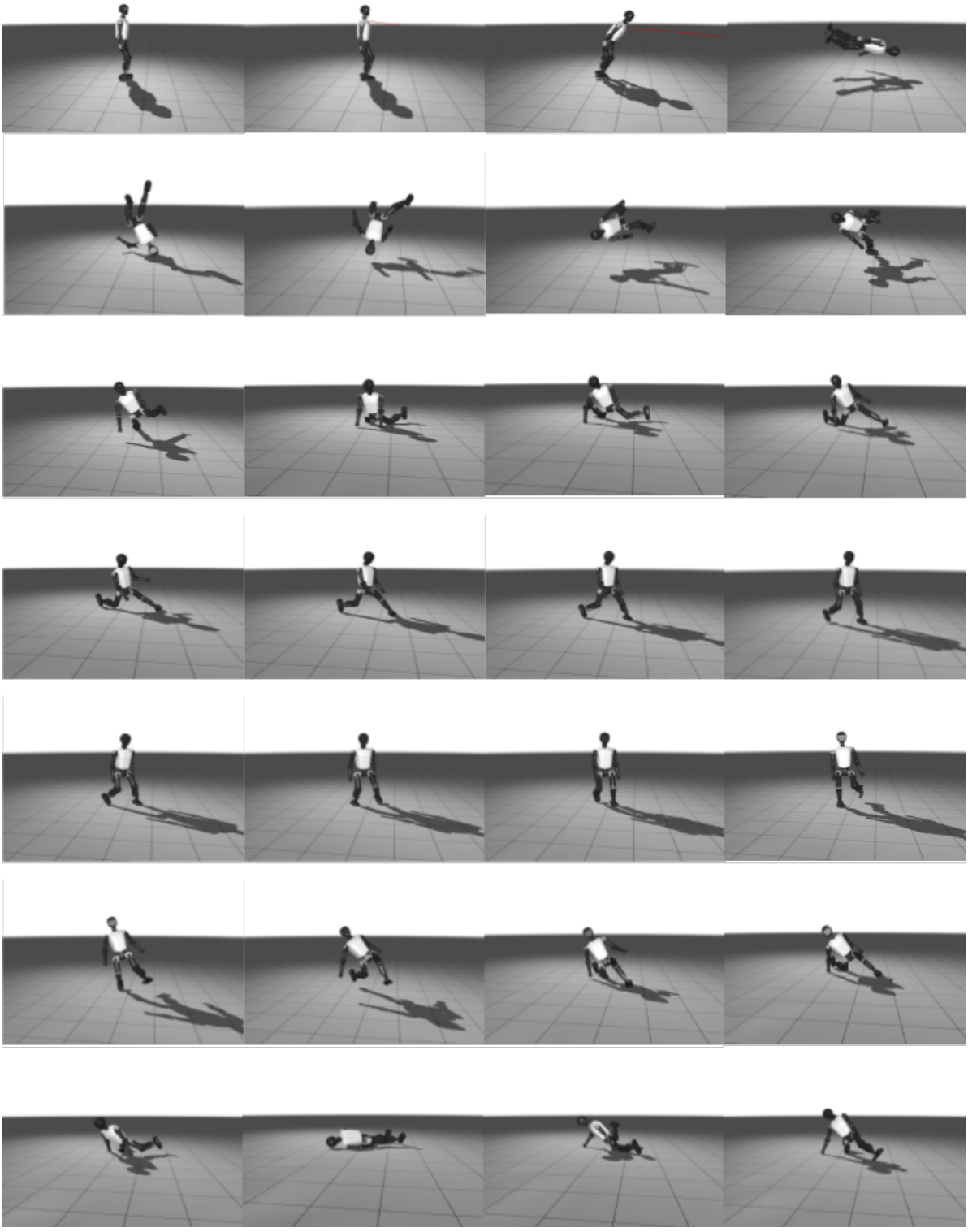

Figure C.11: (a) DAgger-100k forward push evaluation 11

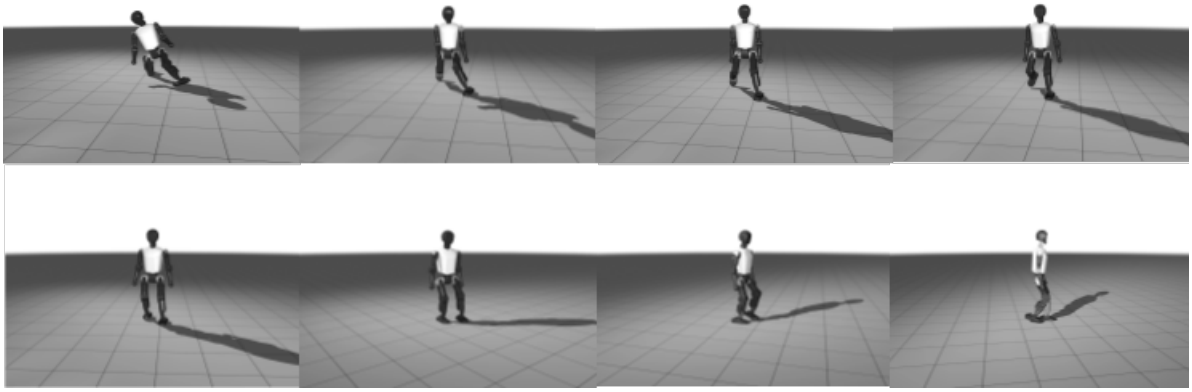

Figure C.12: (b) DAgger-100k forward push evaluation 11

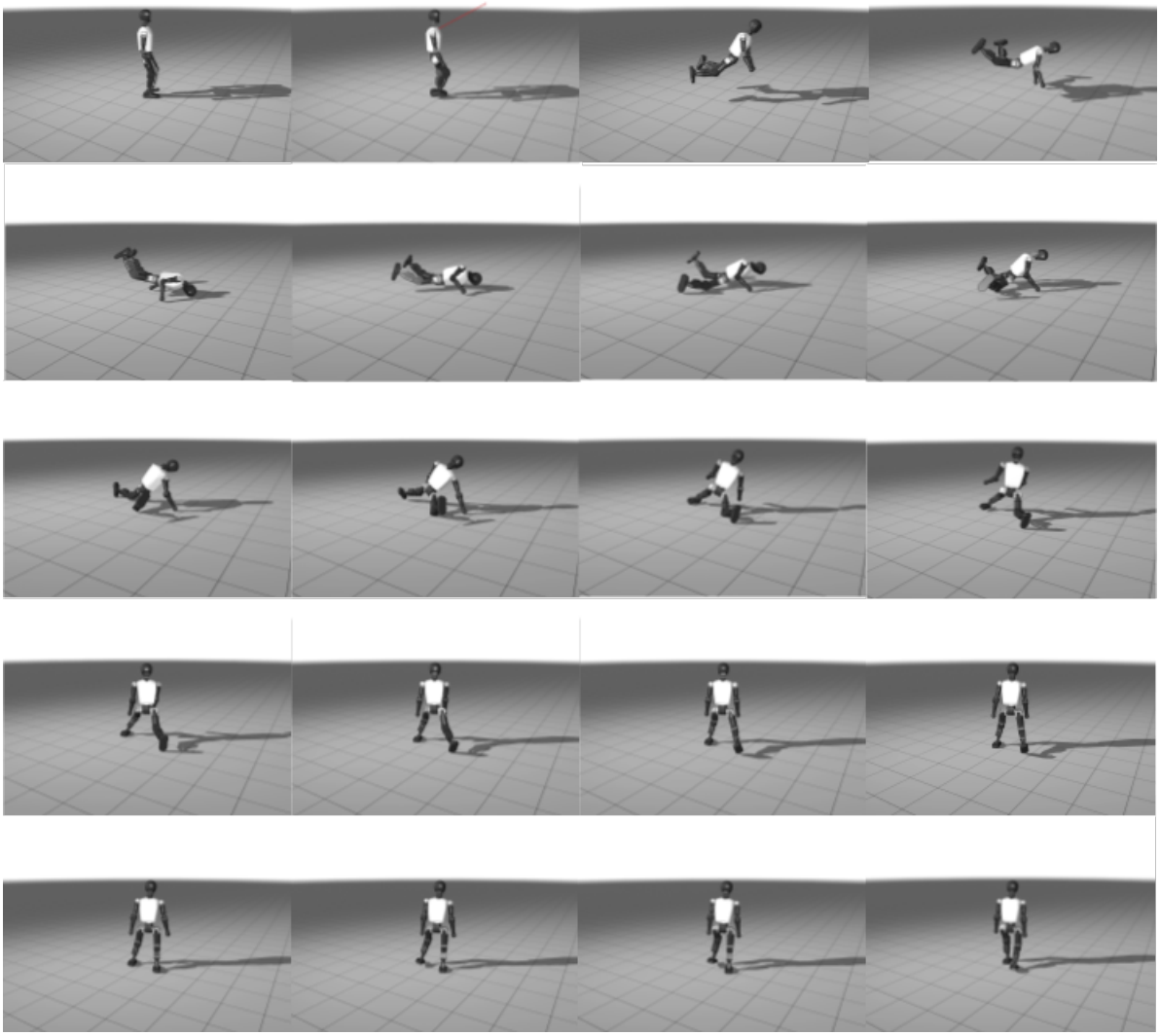

Figure C.13: DAgger-100k forward push evaluation 12

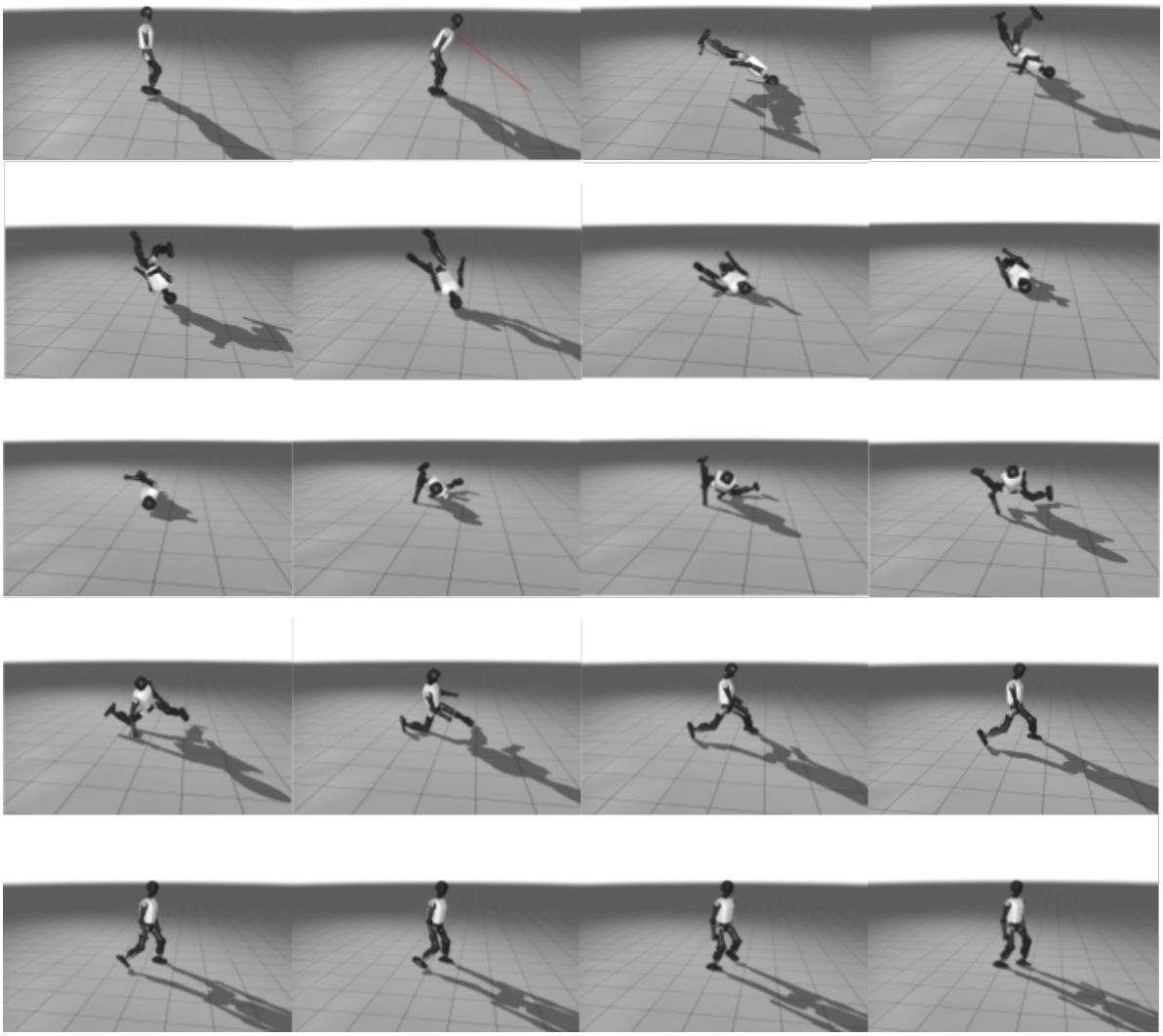

Figure C.14: DAgger-100k forward push evaluation 13

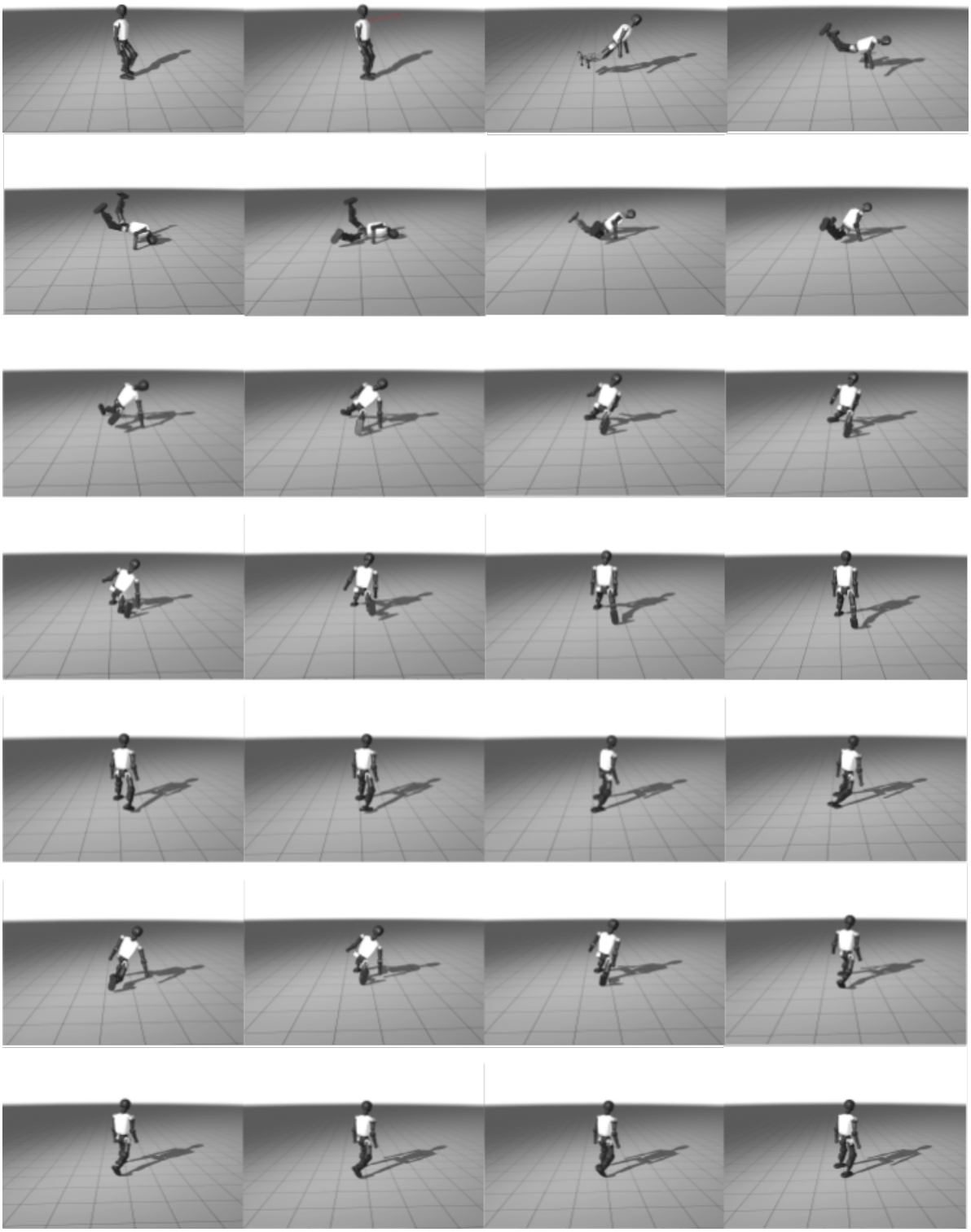

Figure C.15: DAgger-100k forward push evaluation 14

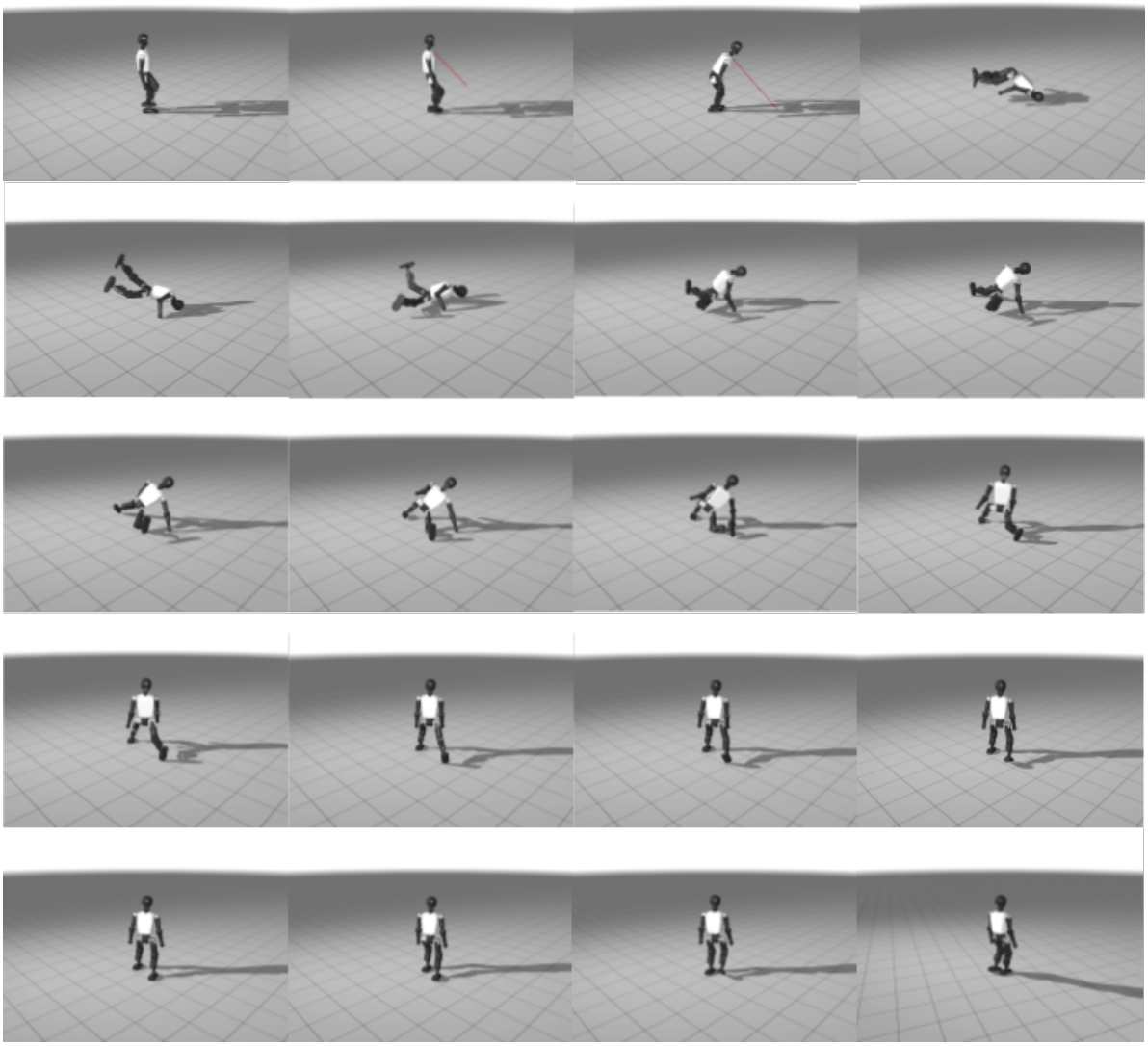

Figure C.16: DAgger-100k forward push evaluation 15

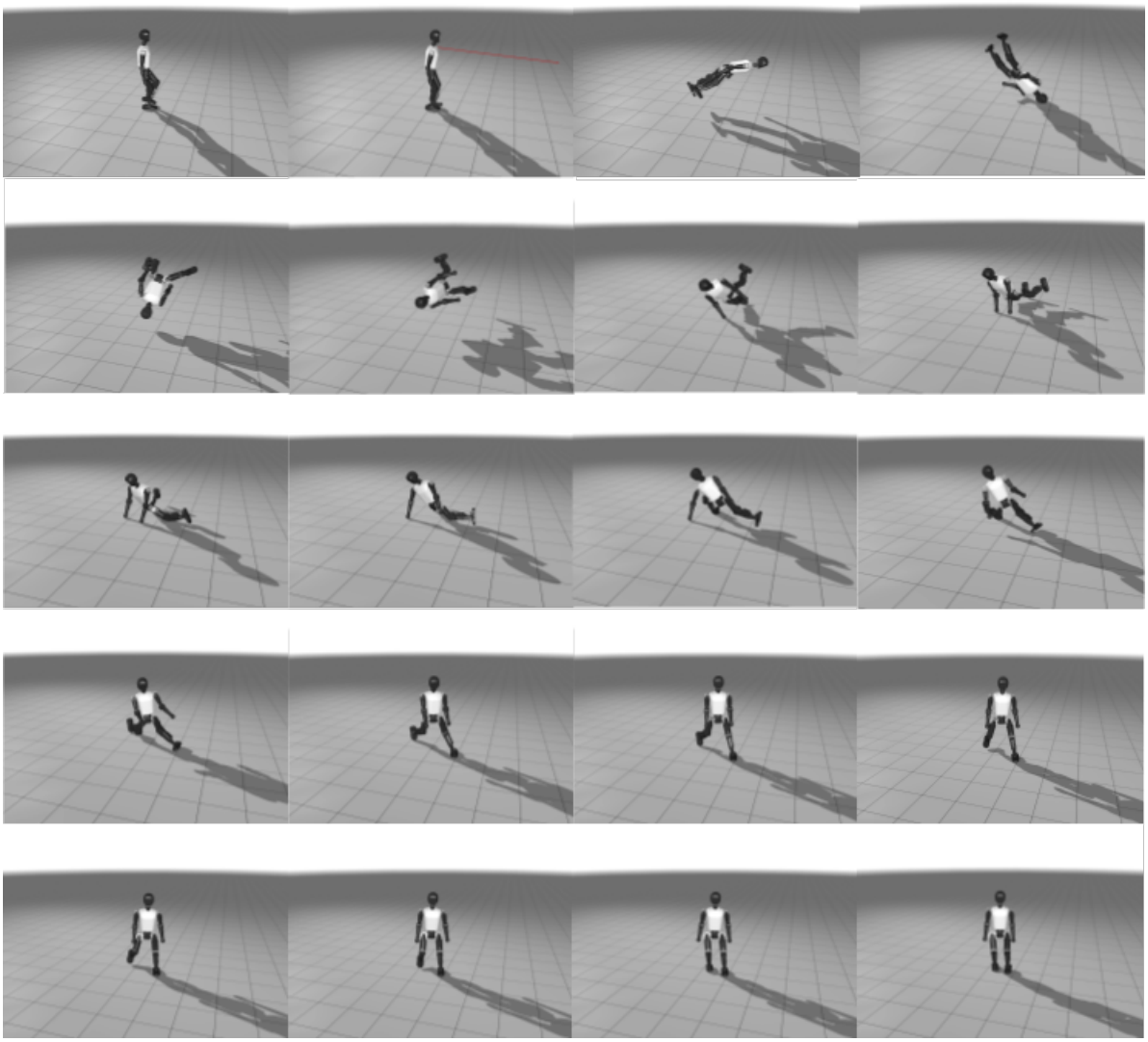

Figure C.17: DAgger-100k forward push evaluation 16

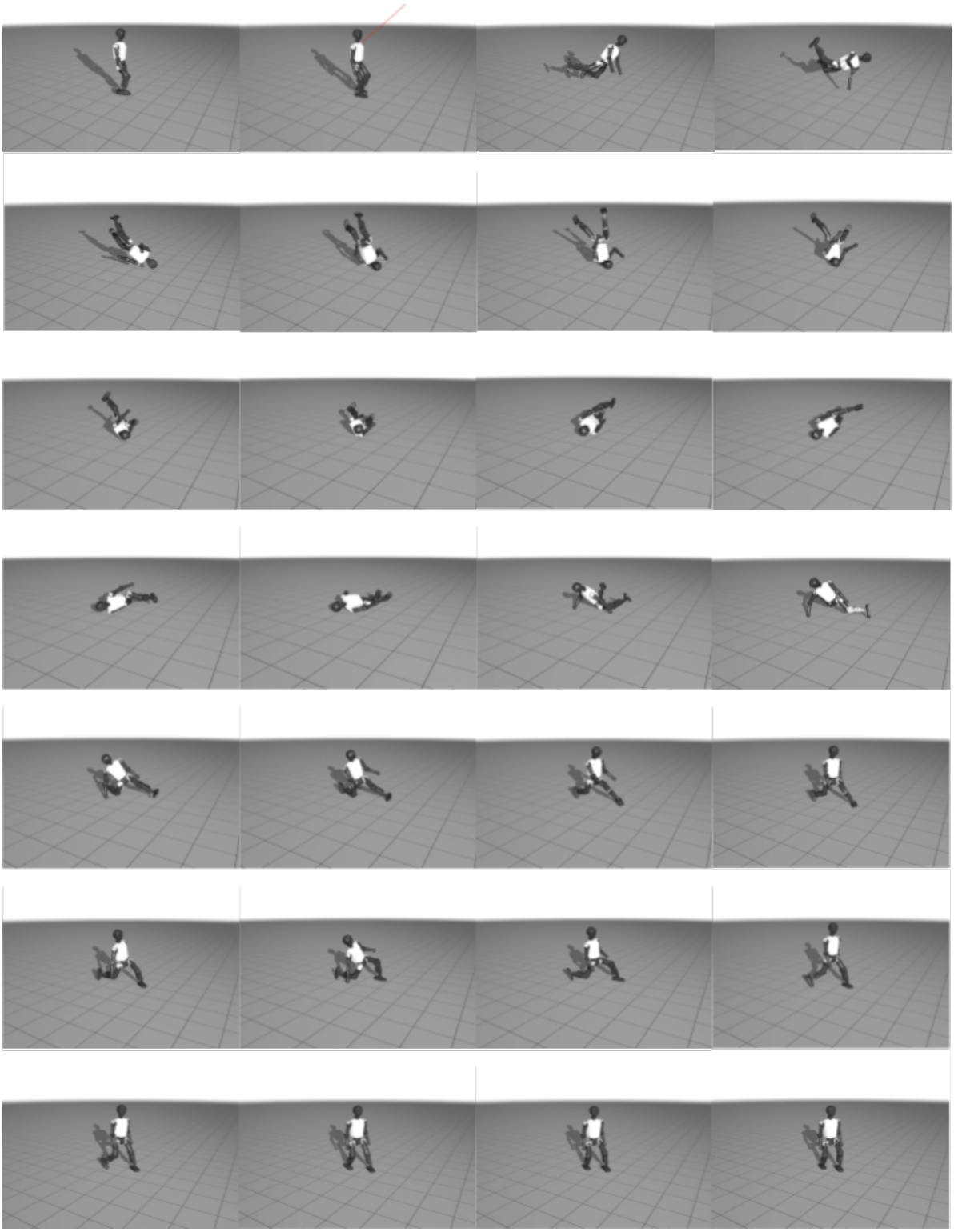

Figure C.18: DAgger-100k forward push evaluation 17

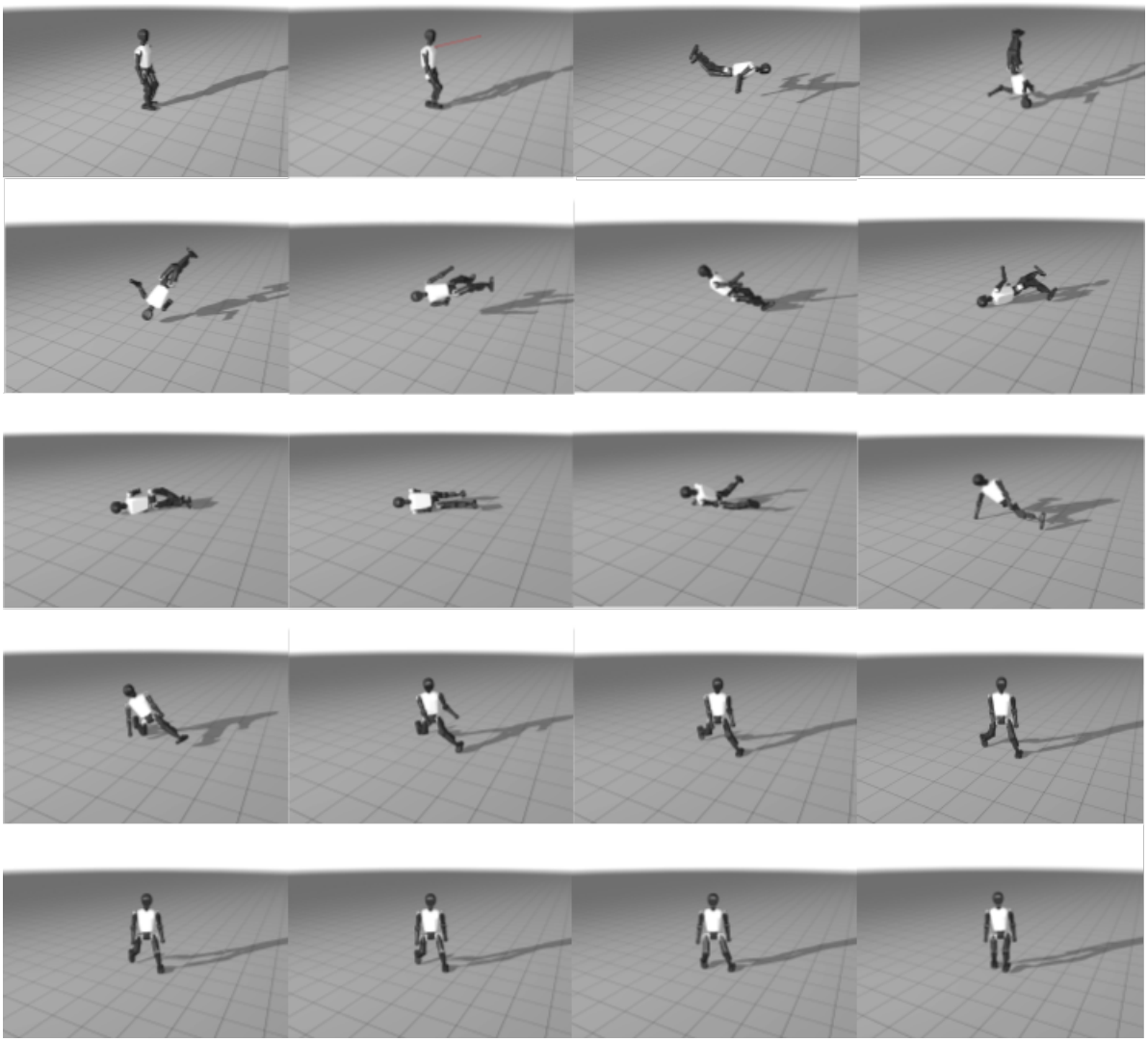

Figure C.19: DAgger-100k forward push evaluation 18

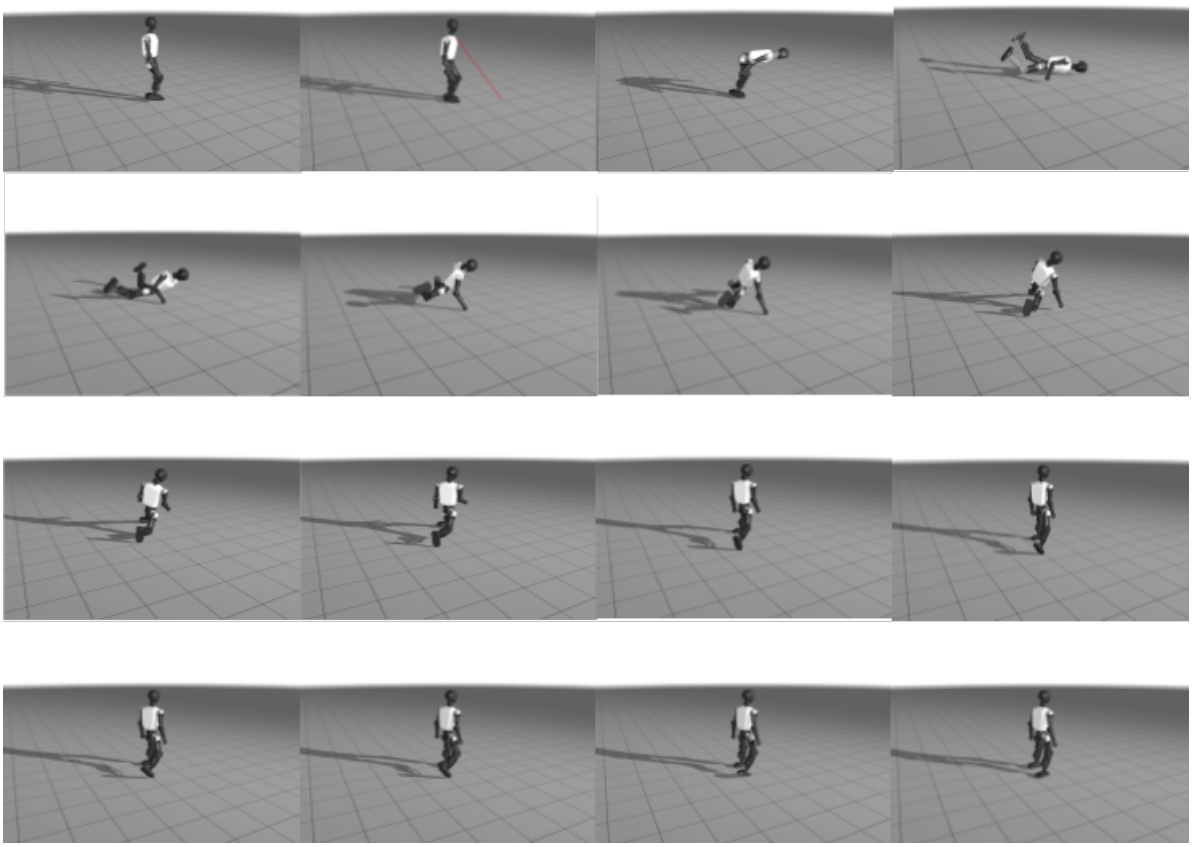

Figure C.20: DAgger-100k forward push evaluation 19

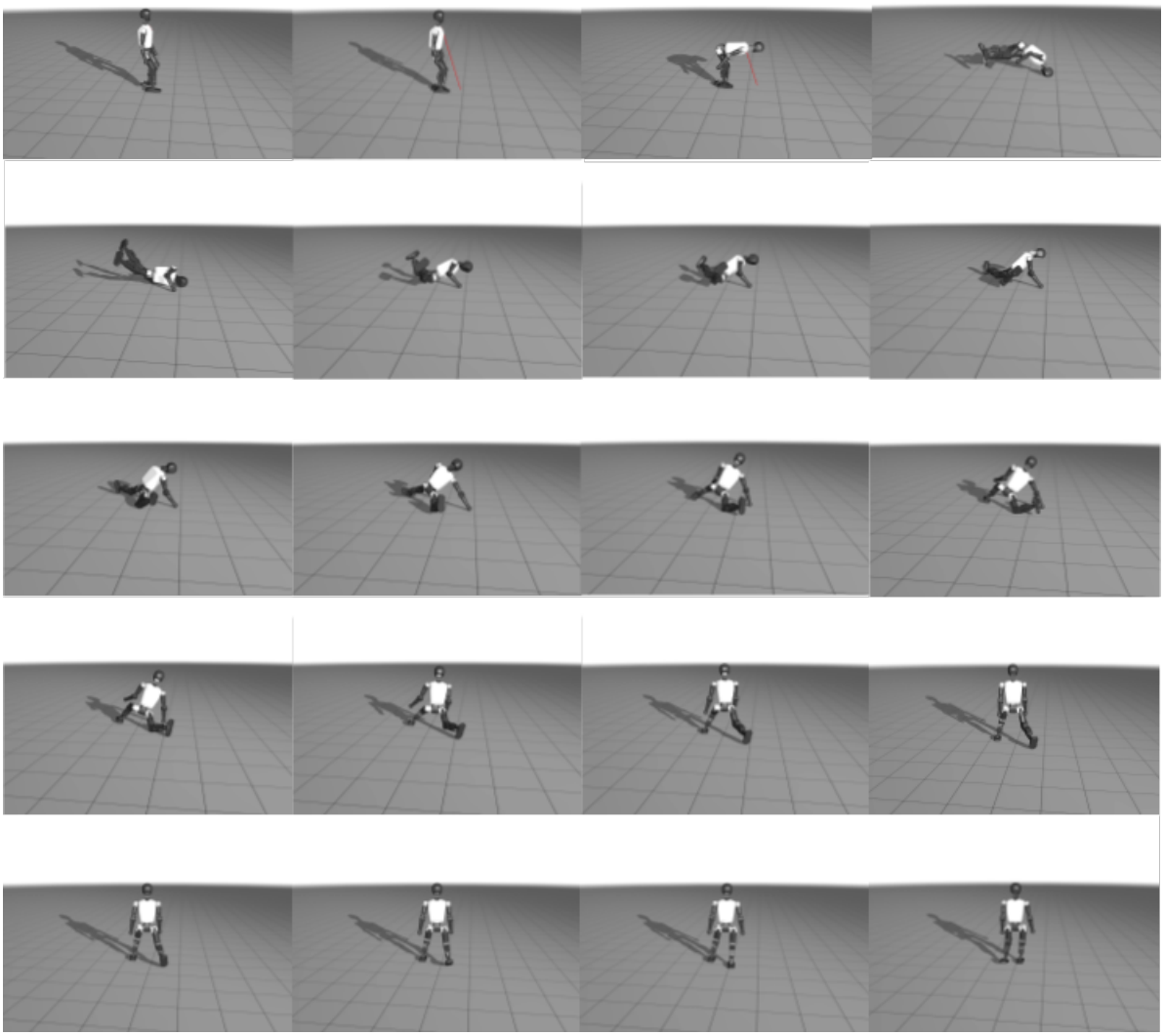

Figure C.21: DAgger-100k forward push evaluation 20

Baseline forward push evaluations

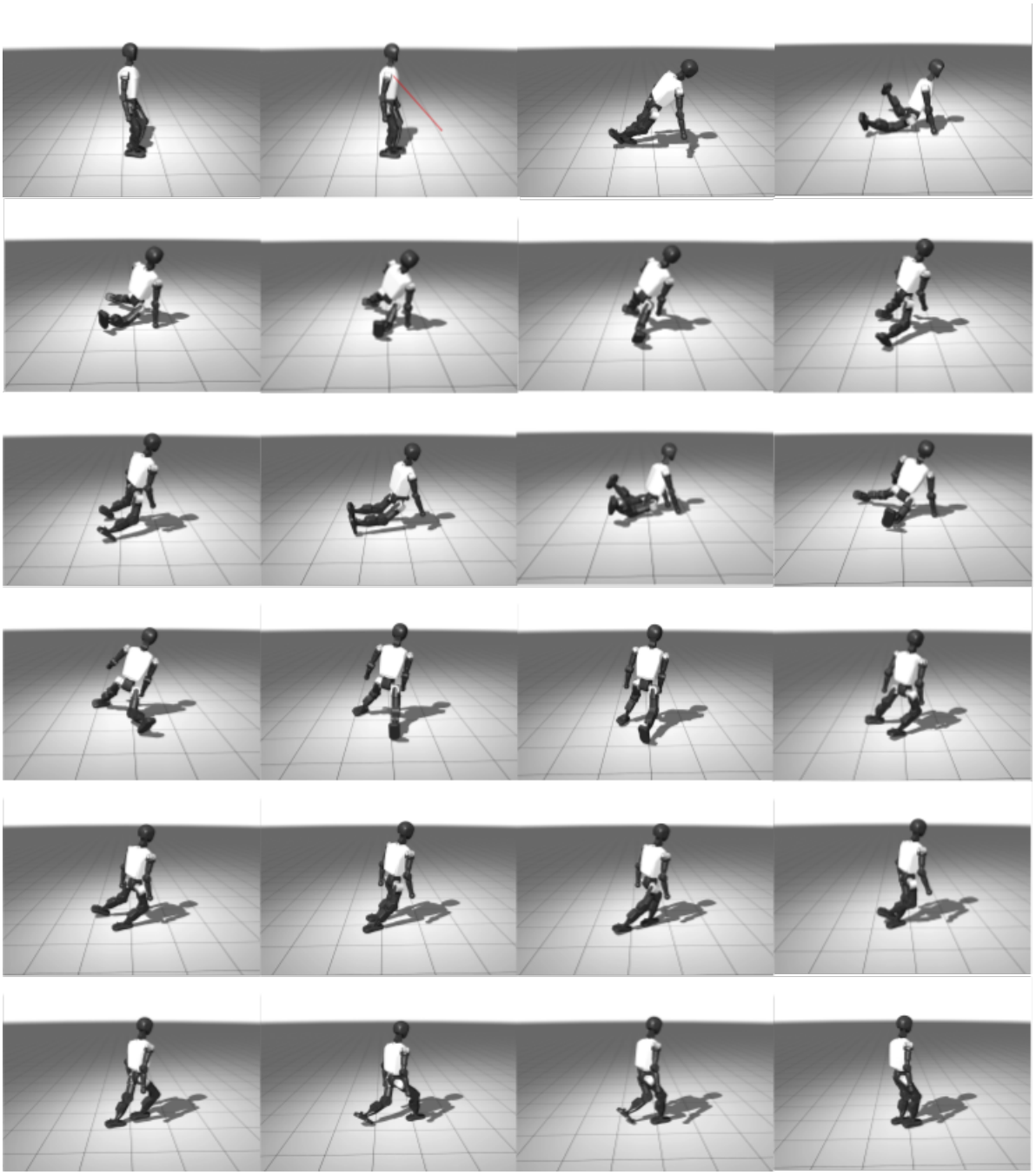

Figure C.22: Baseline forward push evaluation 1

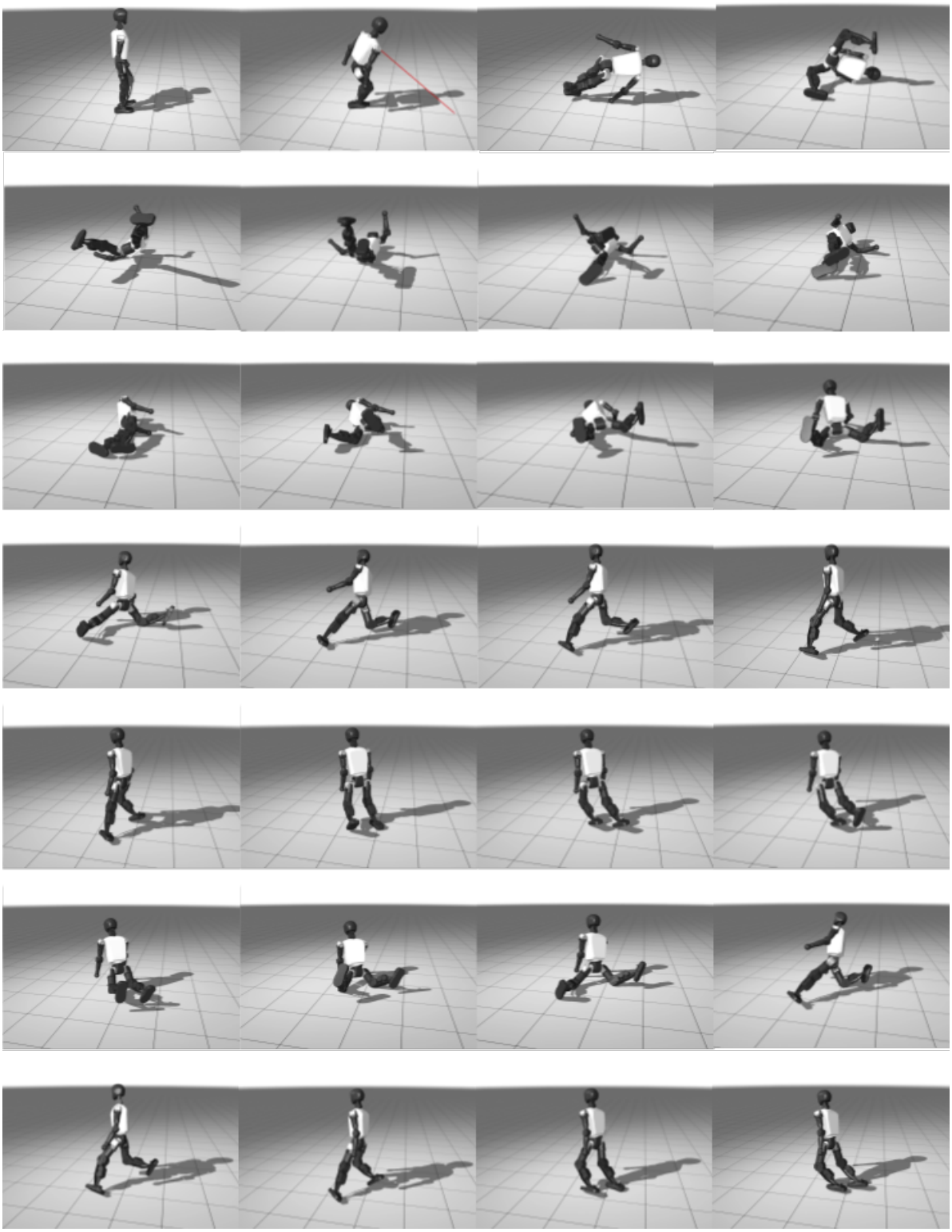

Figure C.23: (a) Baseline forward push evaluation 2

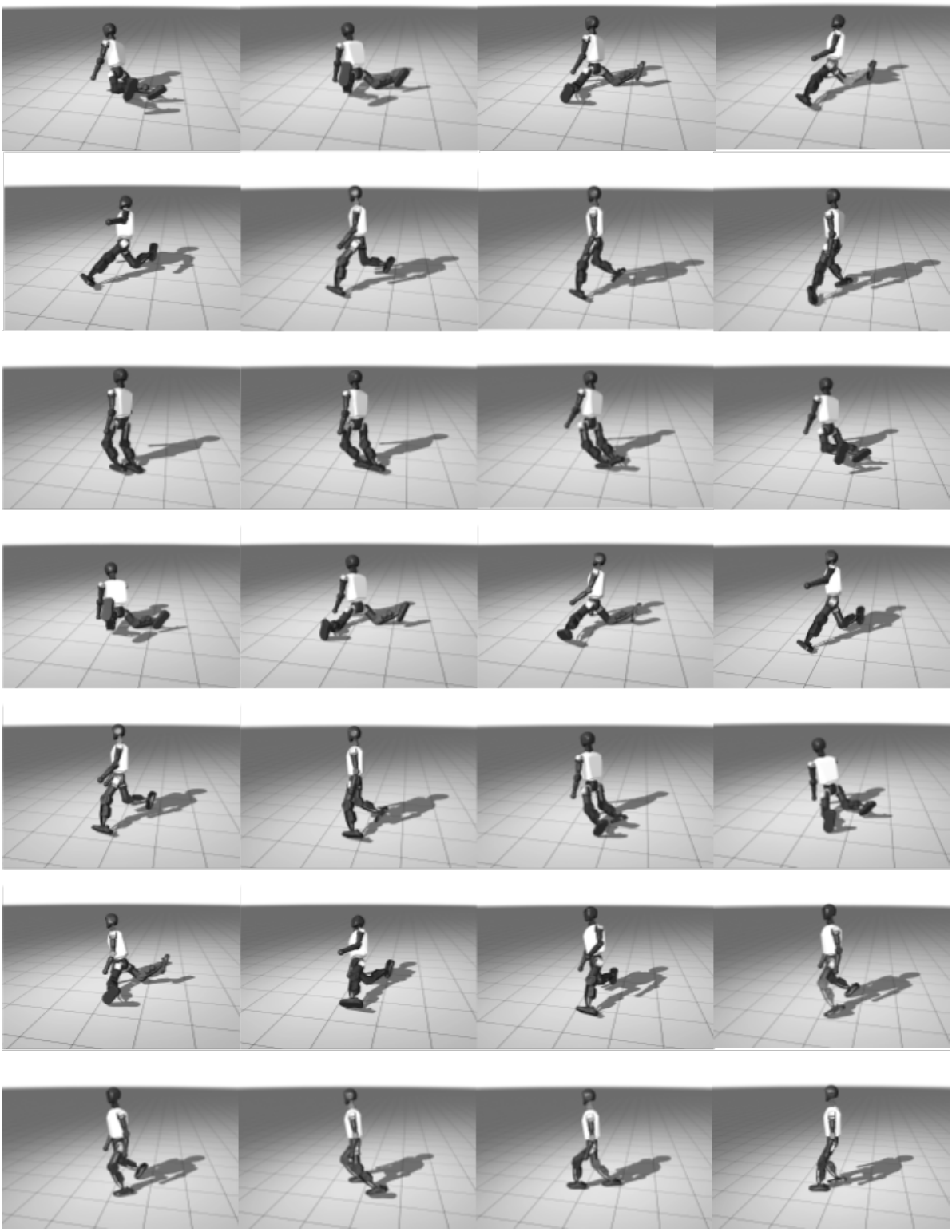

Figure C.24: (b) Baseline forward push evaluation 2

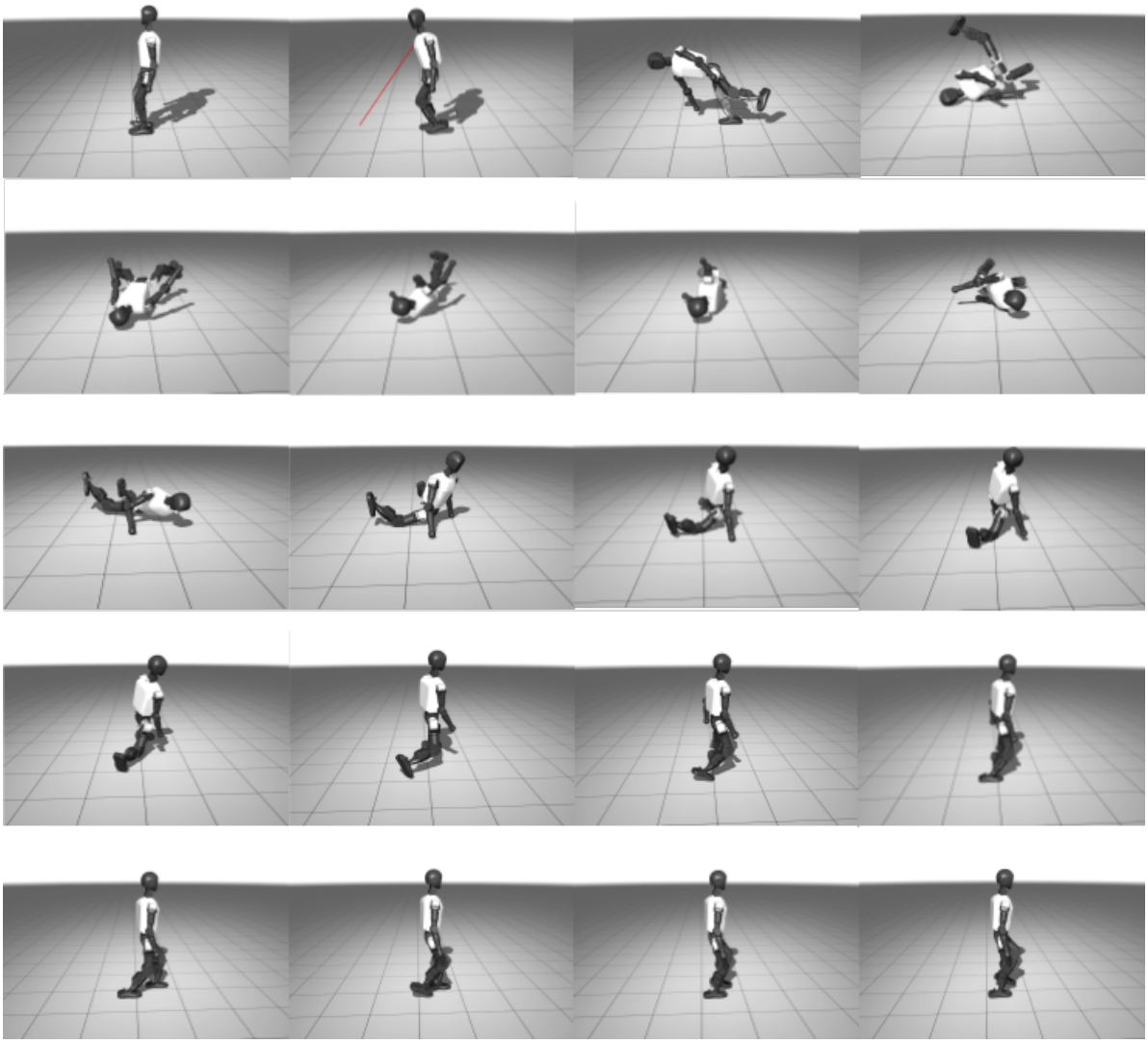

Figure C.25: Baseline forward push evaluation 3

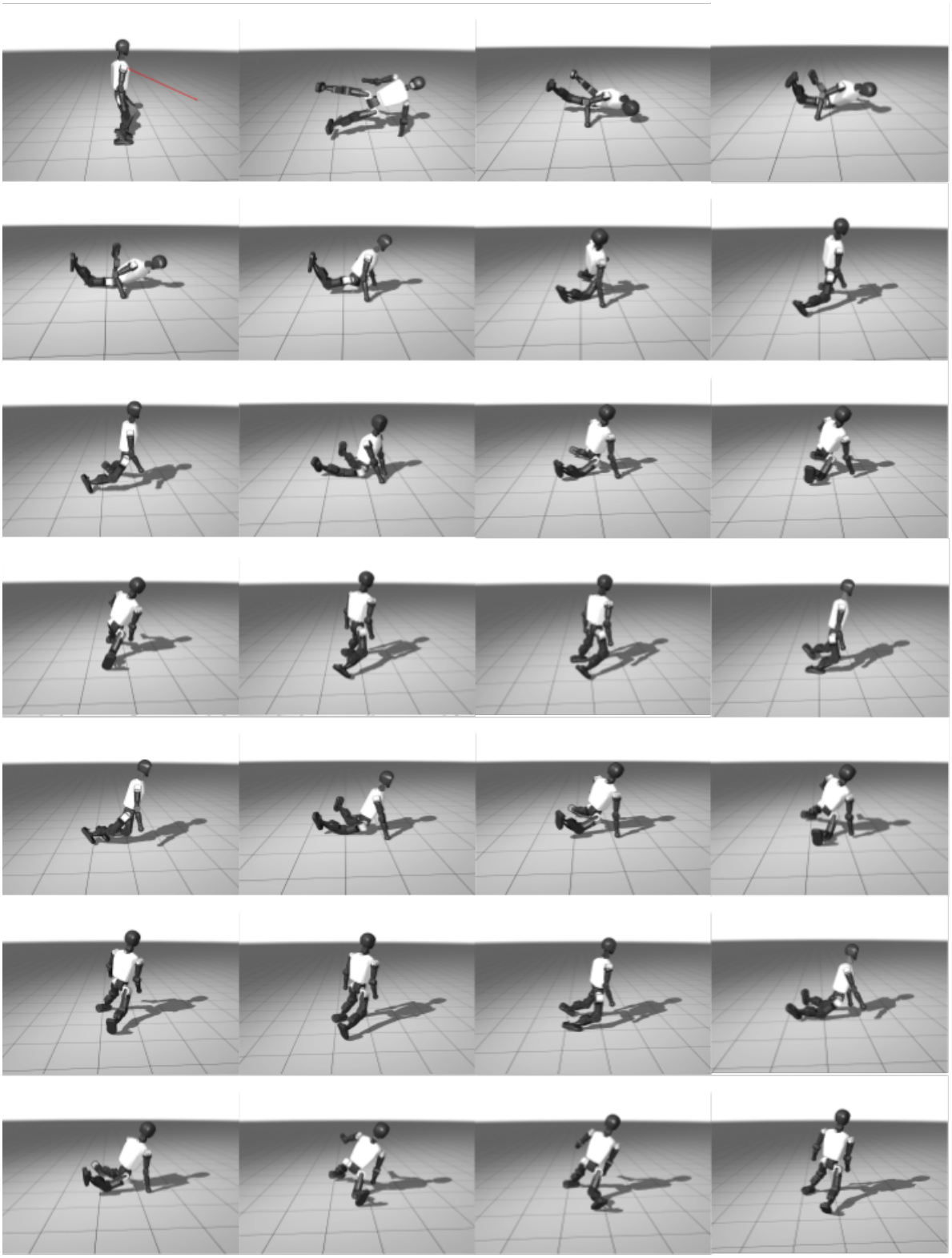

Figure C.26: (a) Baseline forward push evaluation 4

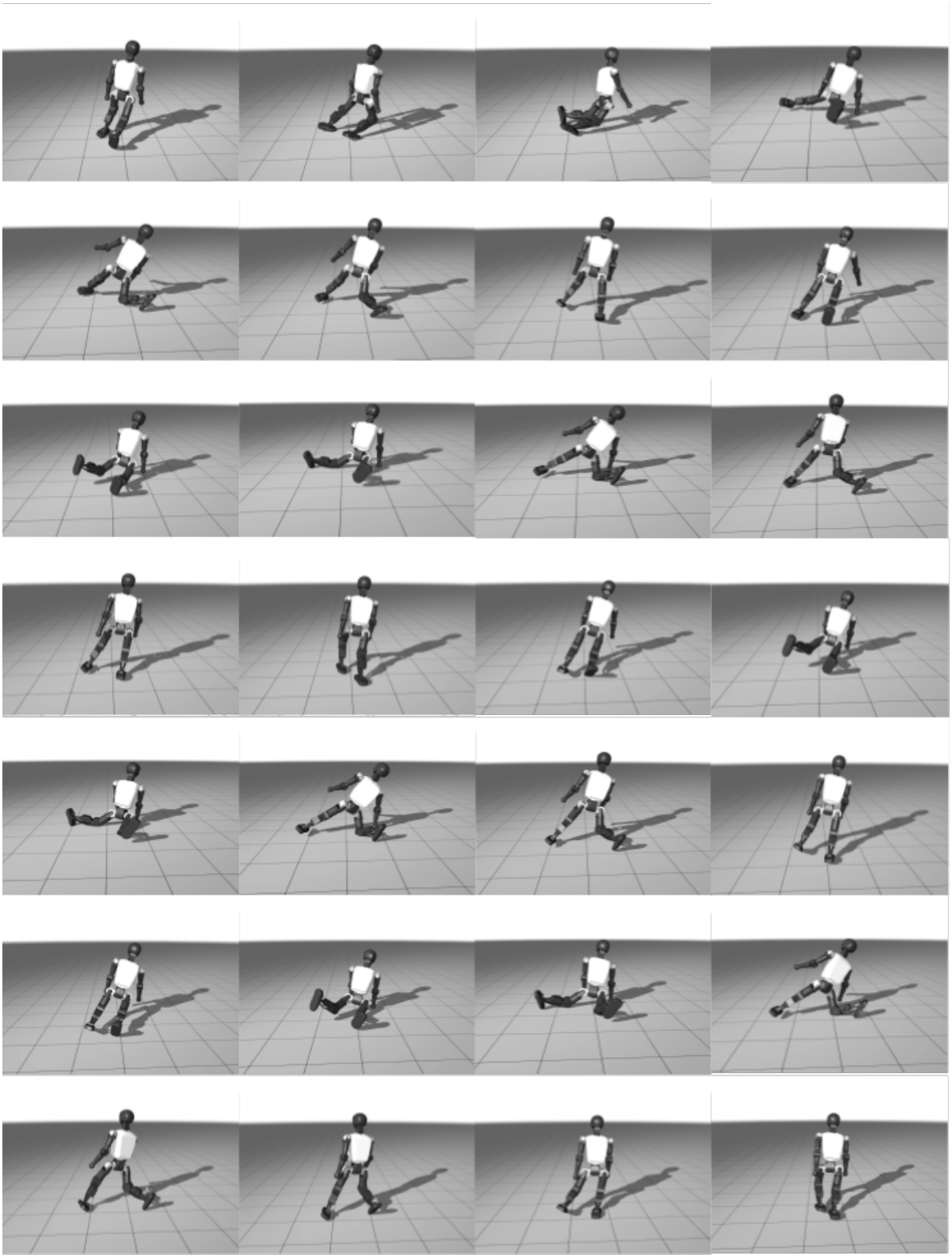

Figure C.27: (b) Baseline forward push evaluation 4

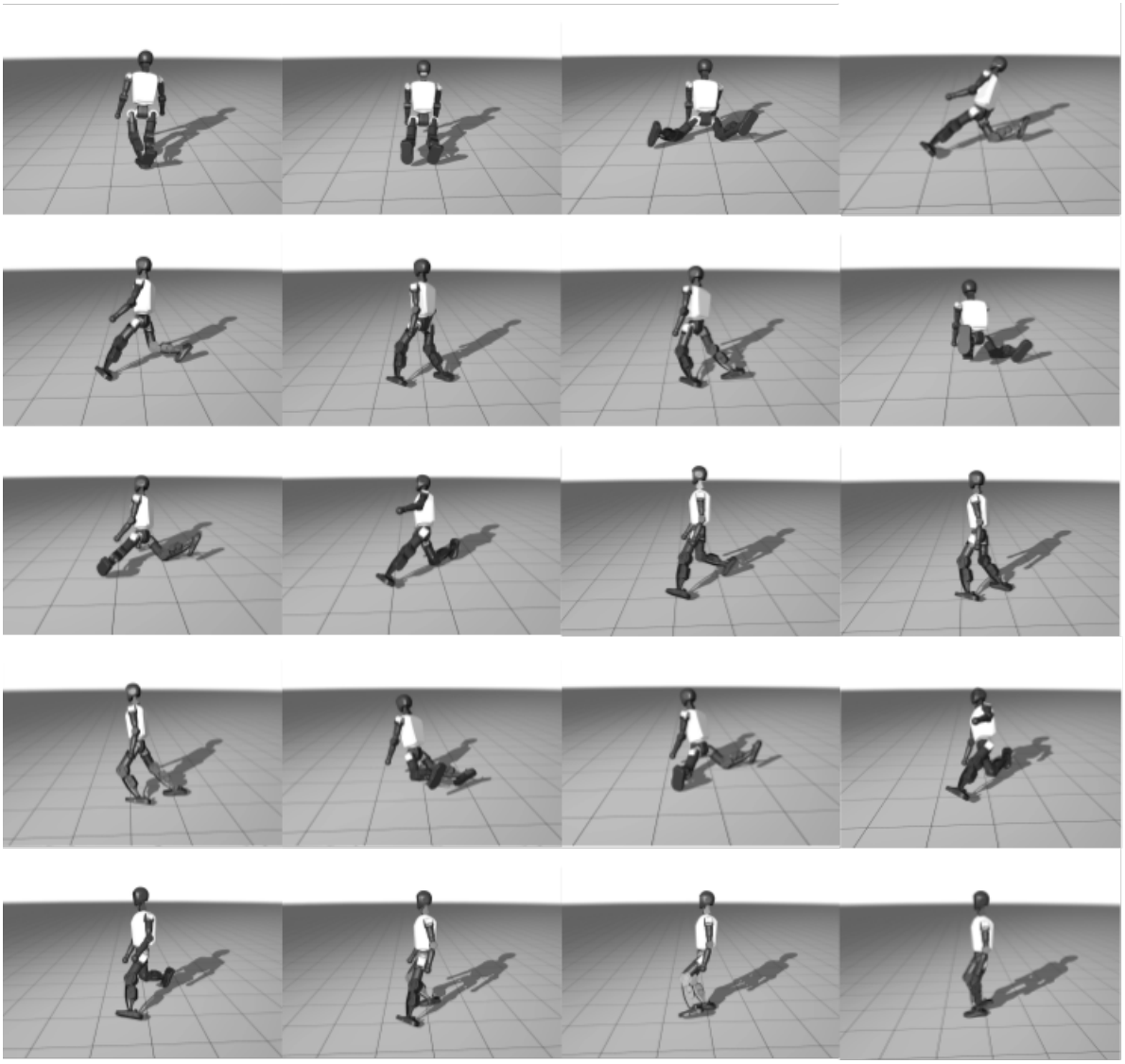

Figure C.28: (c) Baseline forward push evaluation 4

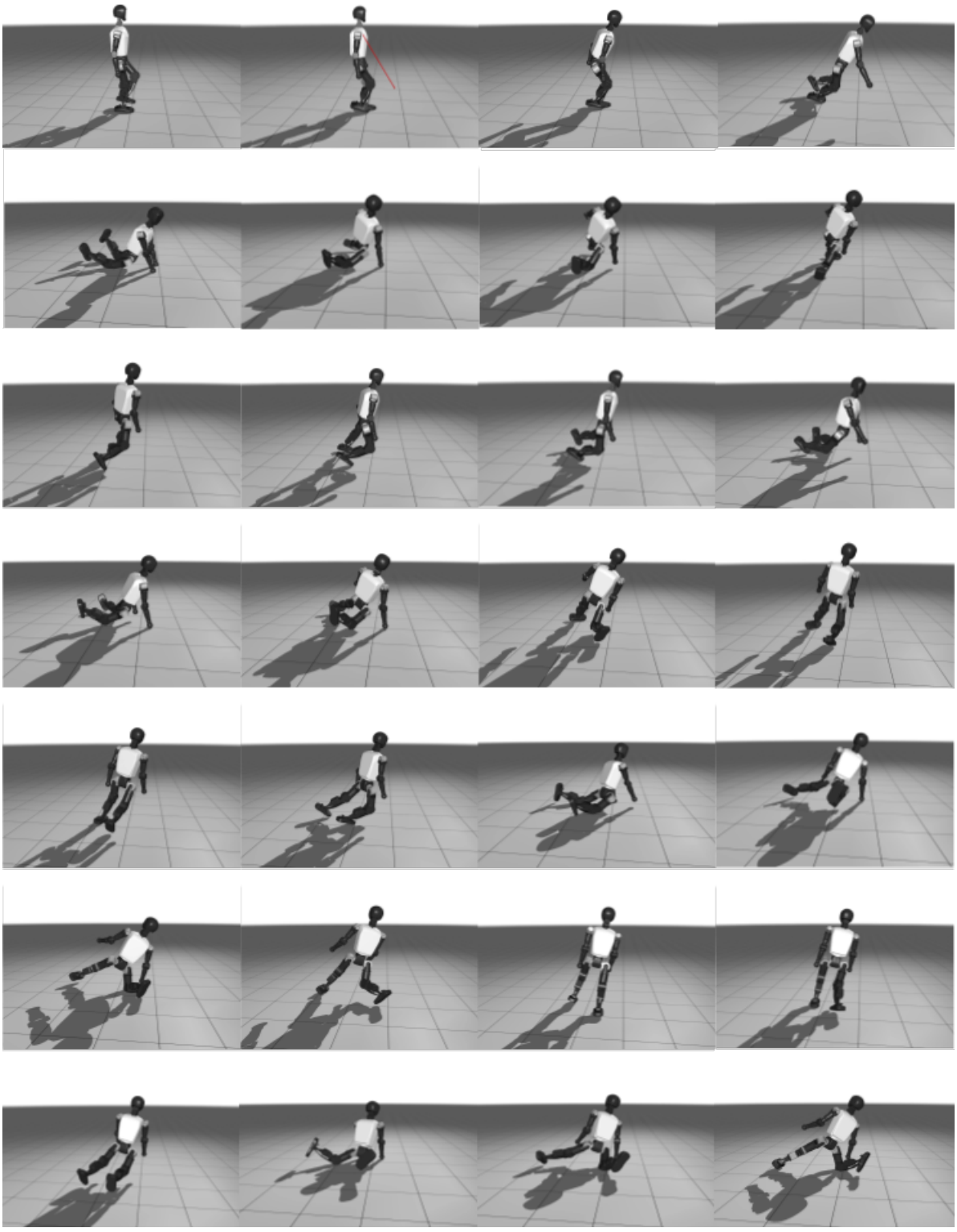

Figure C.29: (a) Baseline forward push evaluation 5

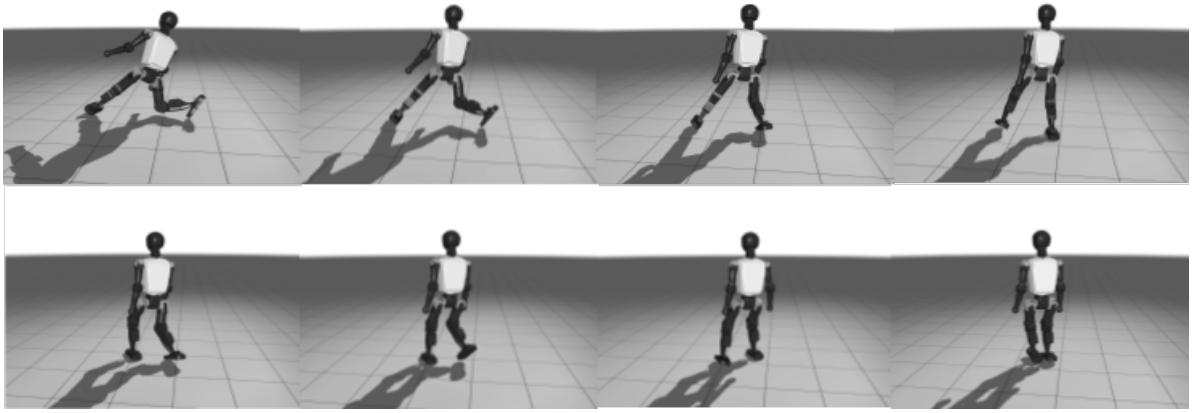

Figure C.30: (b) Baseline forward push evaluation 5

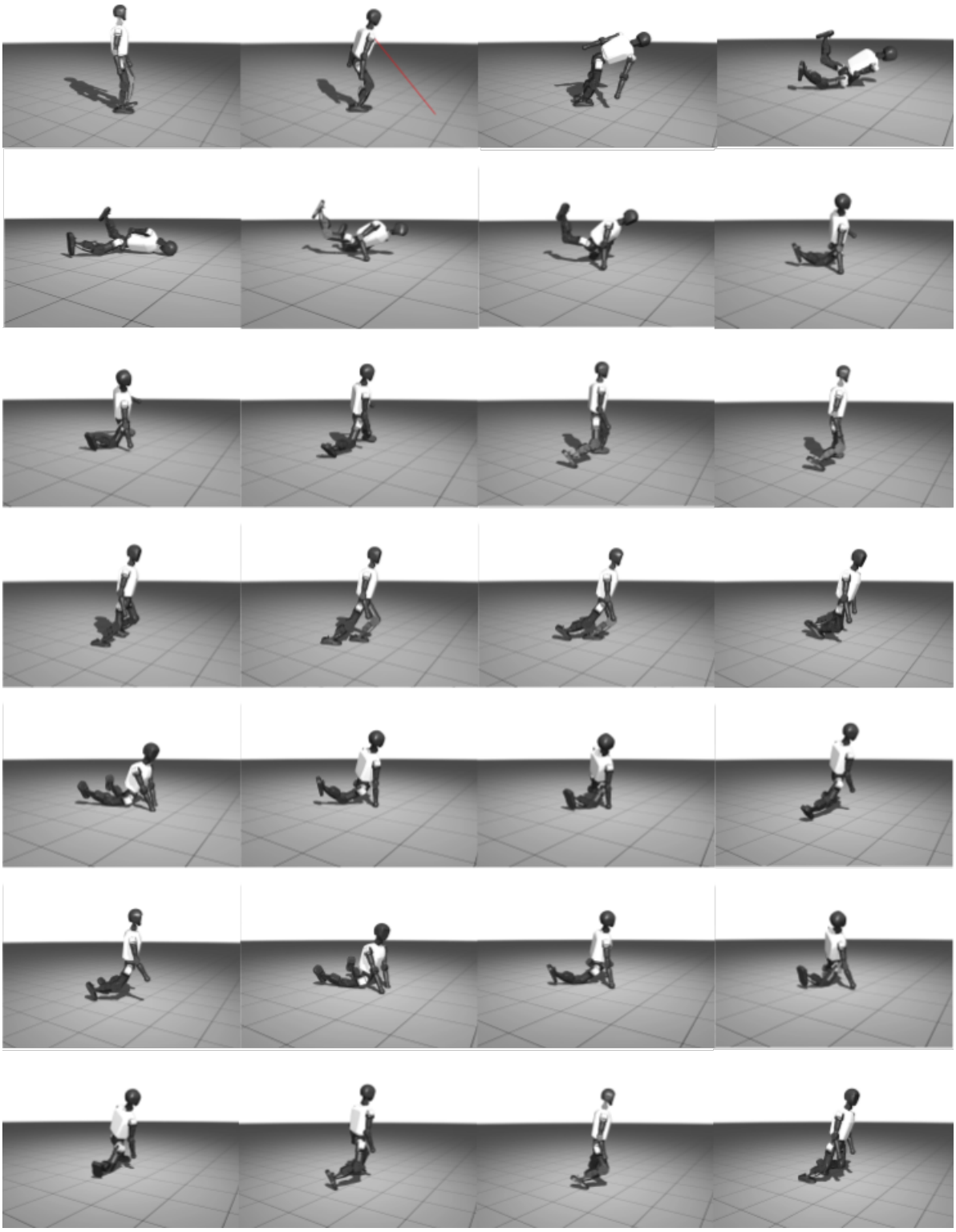

Figure C.31: (a) Baseline forward push evaluation 6

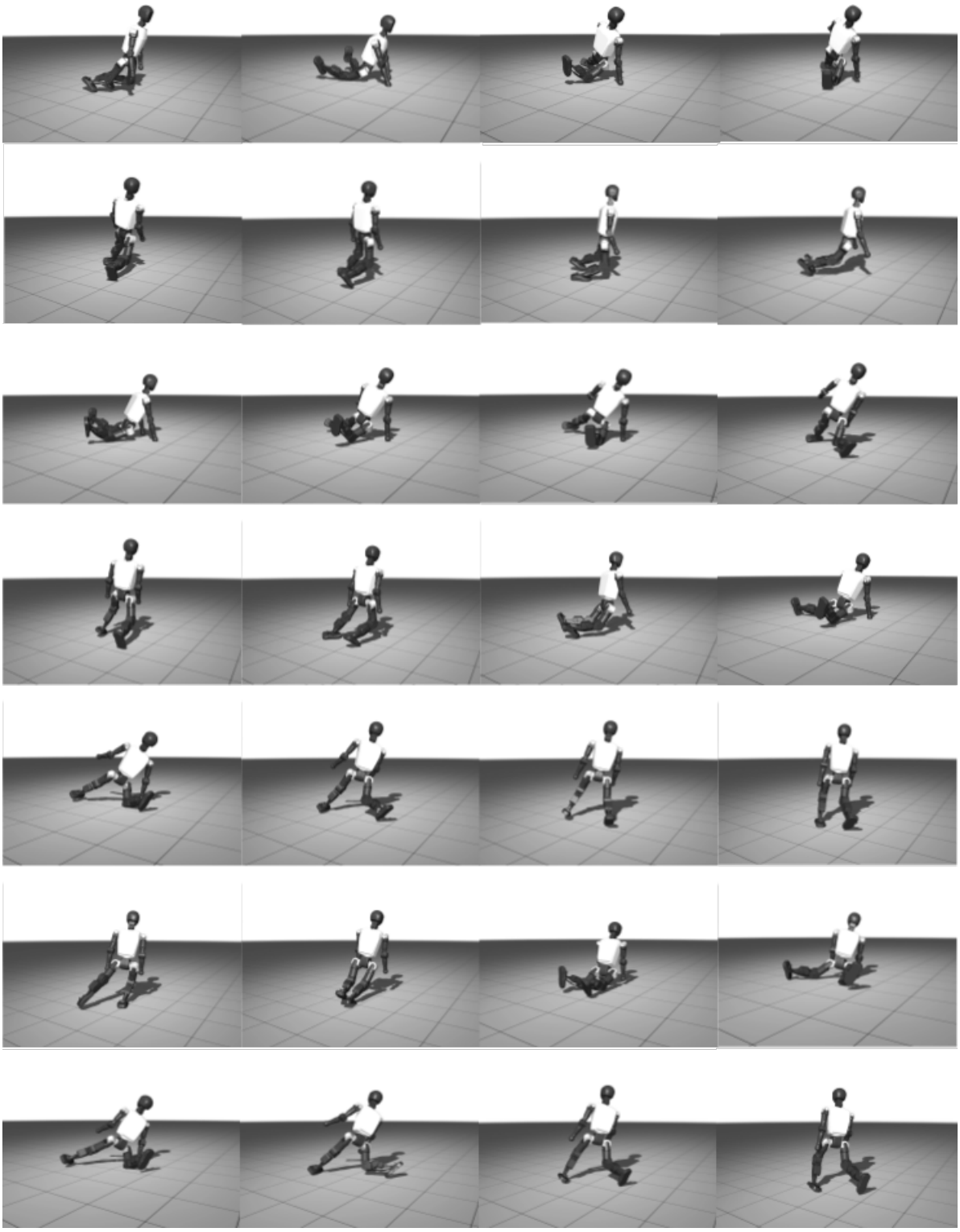

Figure C.32: (b) Baseline forward push evaluation 6

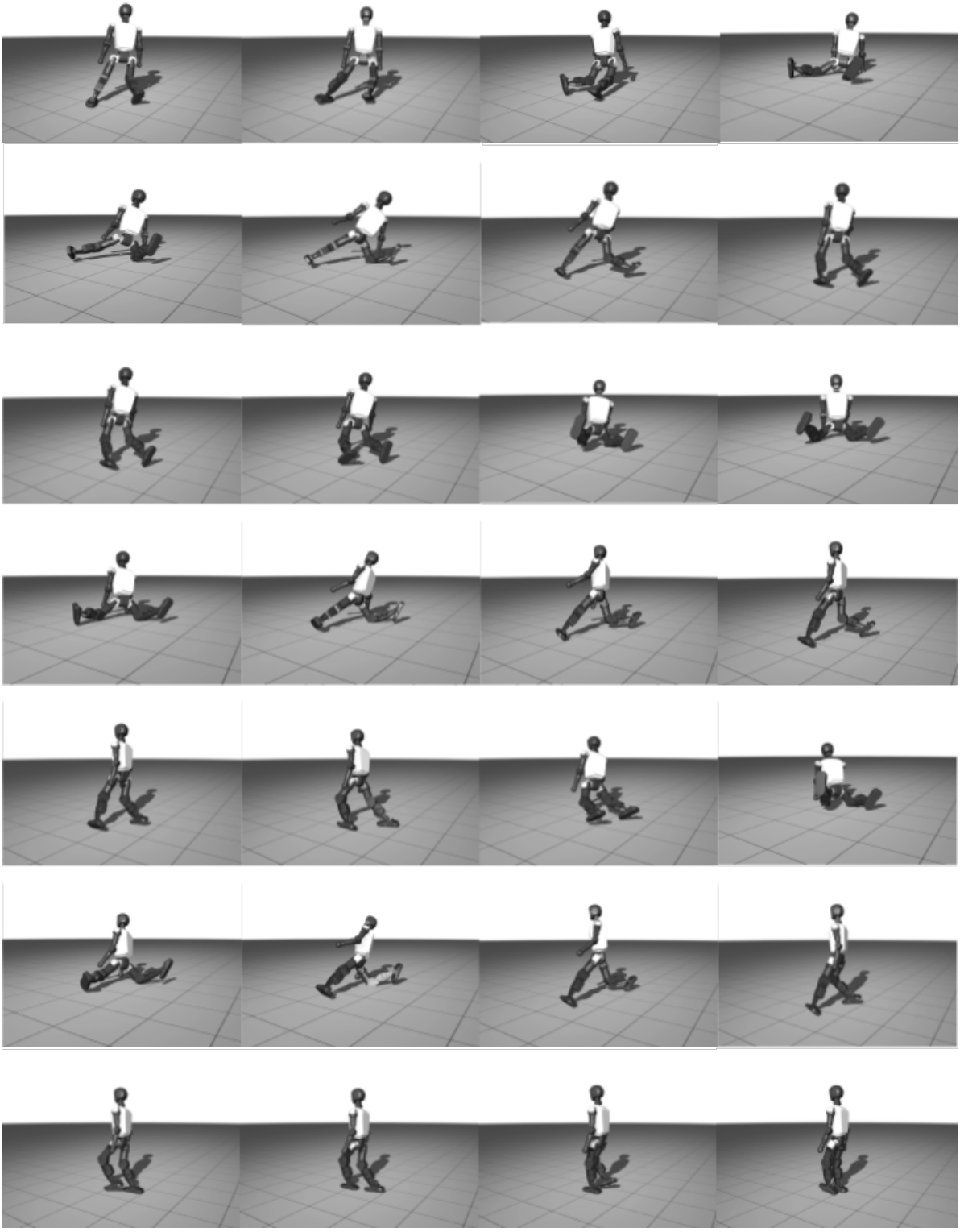

Figure C.33: (c) Baseline forward push evaluation 6

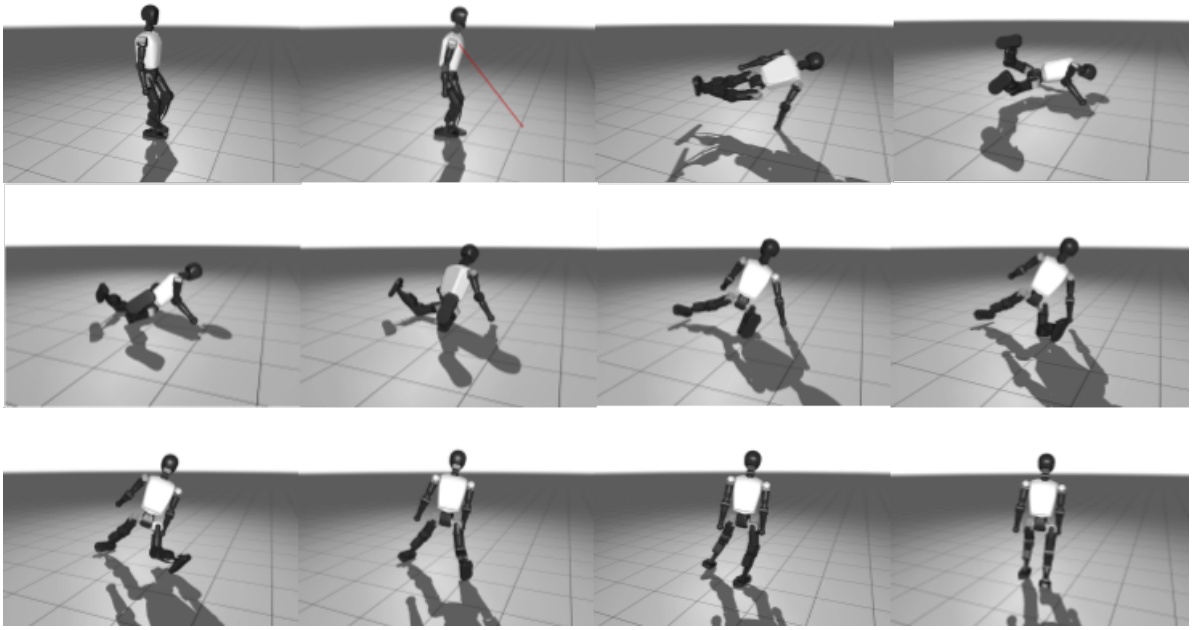

Figure C.34: Baseline forward push evaluation 7

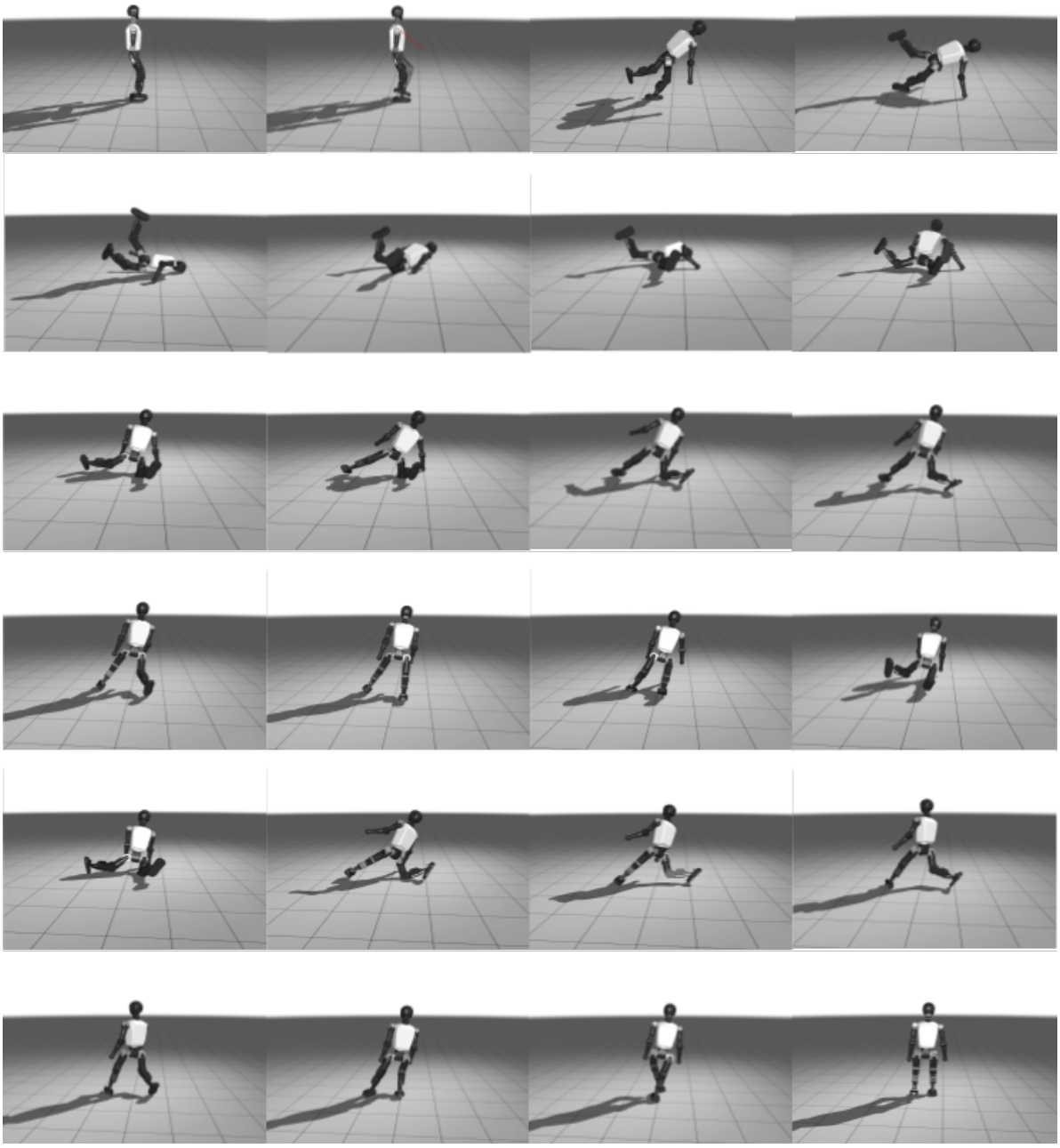

Figure C.35: Baseline forward push evaluation 8

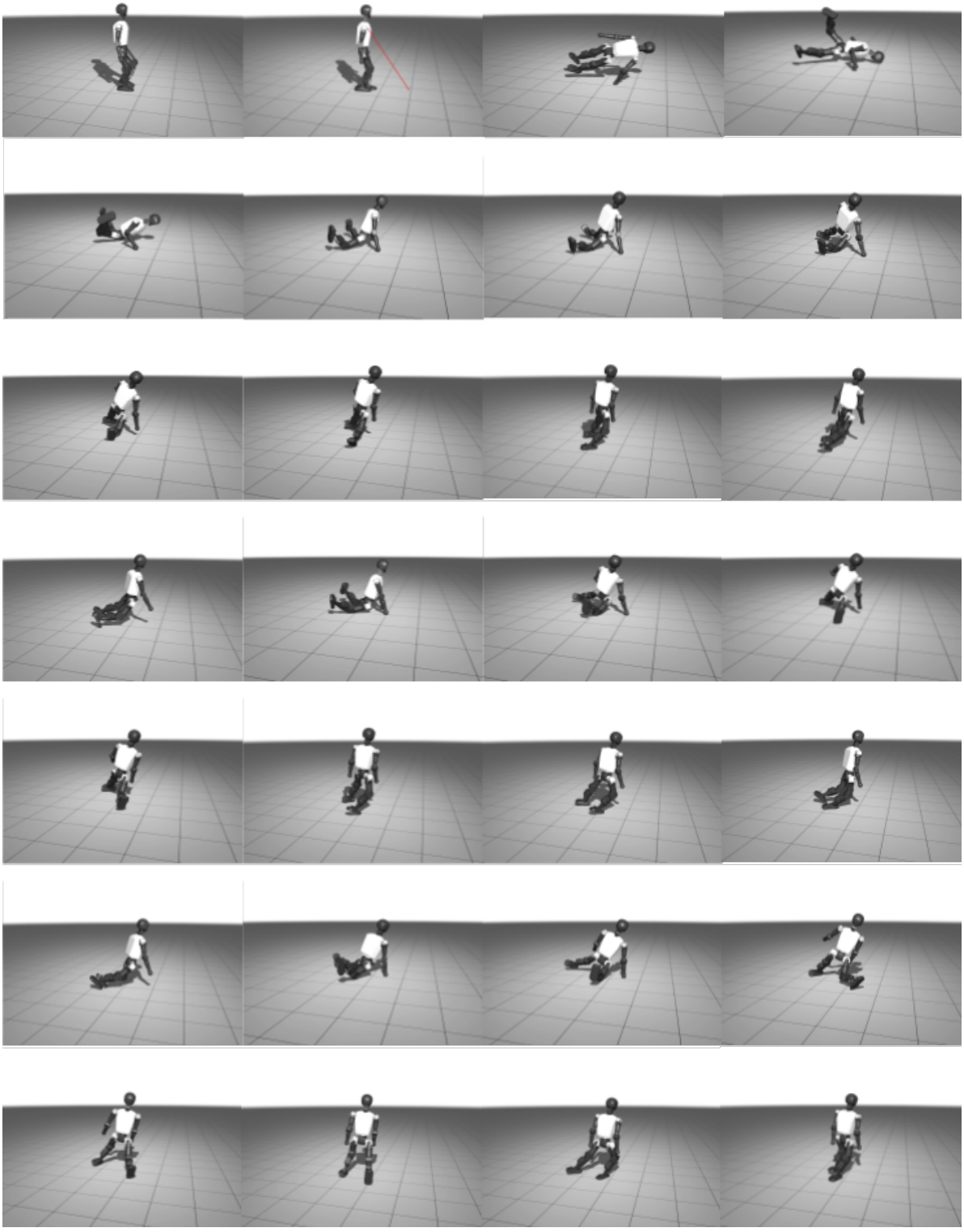

Figure C.36: (a) Baseline forward push evaluation 9

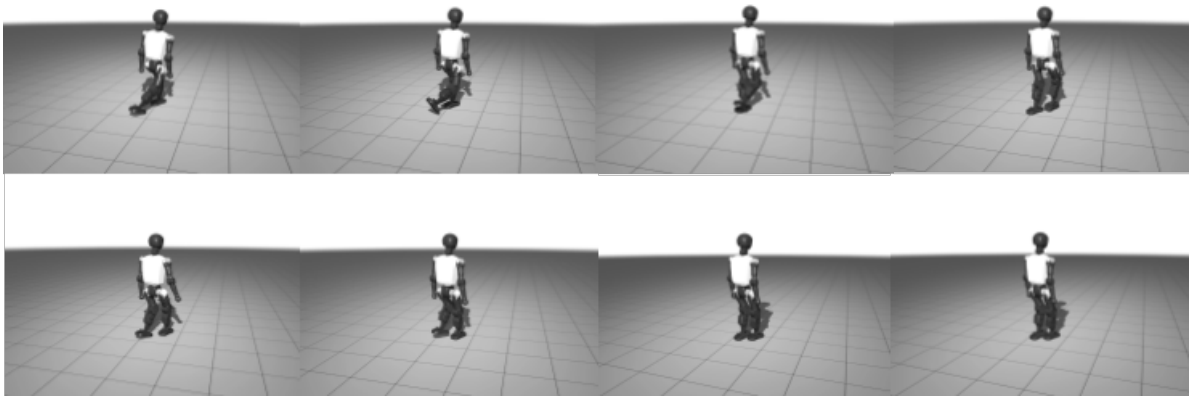

Figure C.37: (b) Baseline forward push evaluation 9

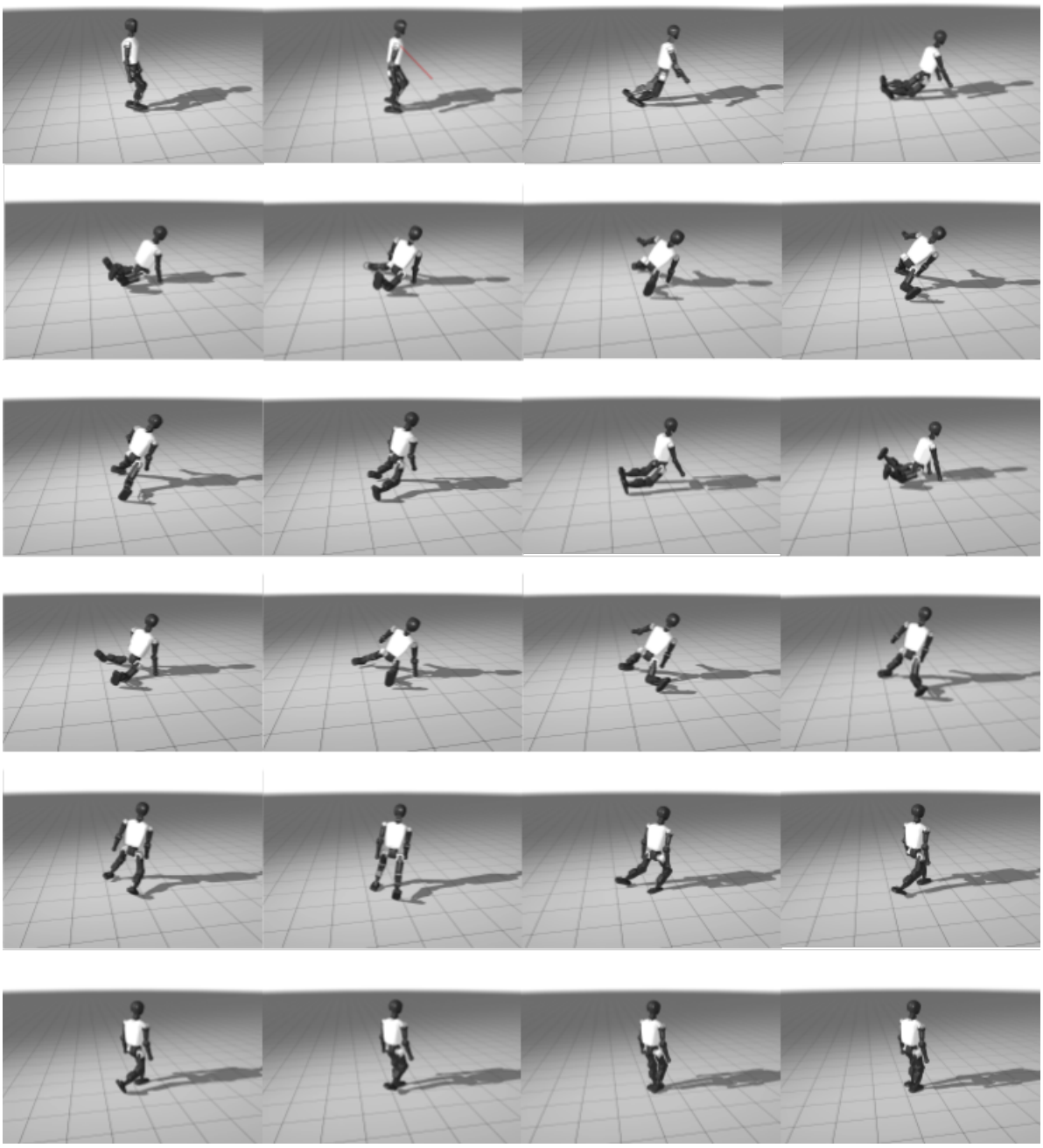

Figure C.38: Baseline forward push evaluation 10

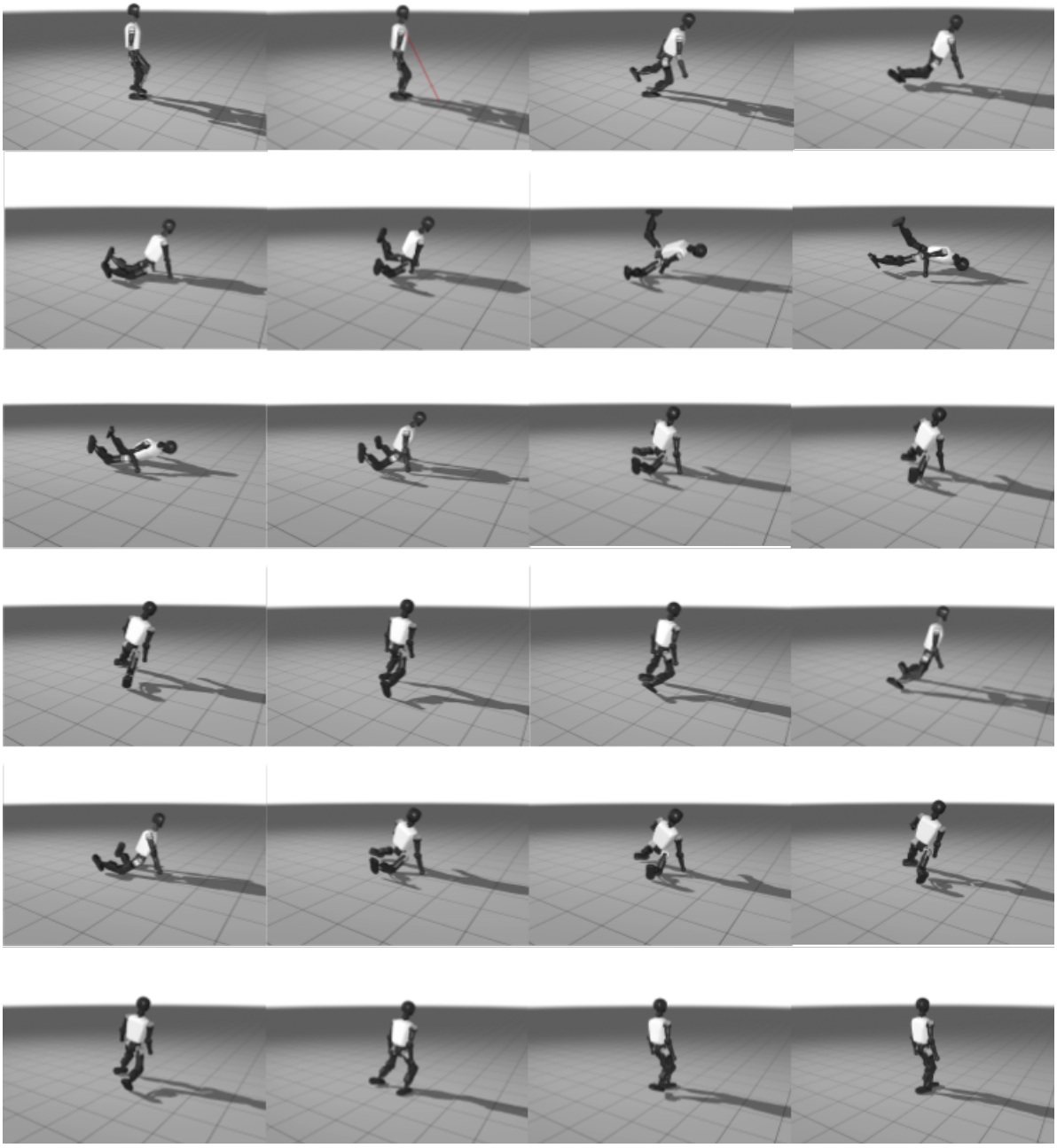

Figure C.39: Baseline forward push evaluation 11

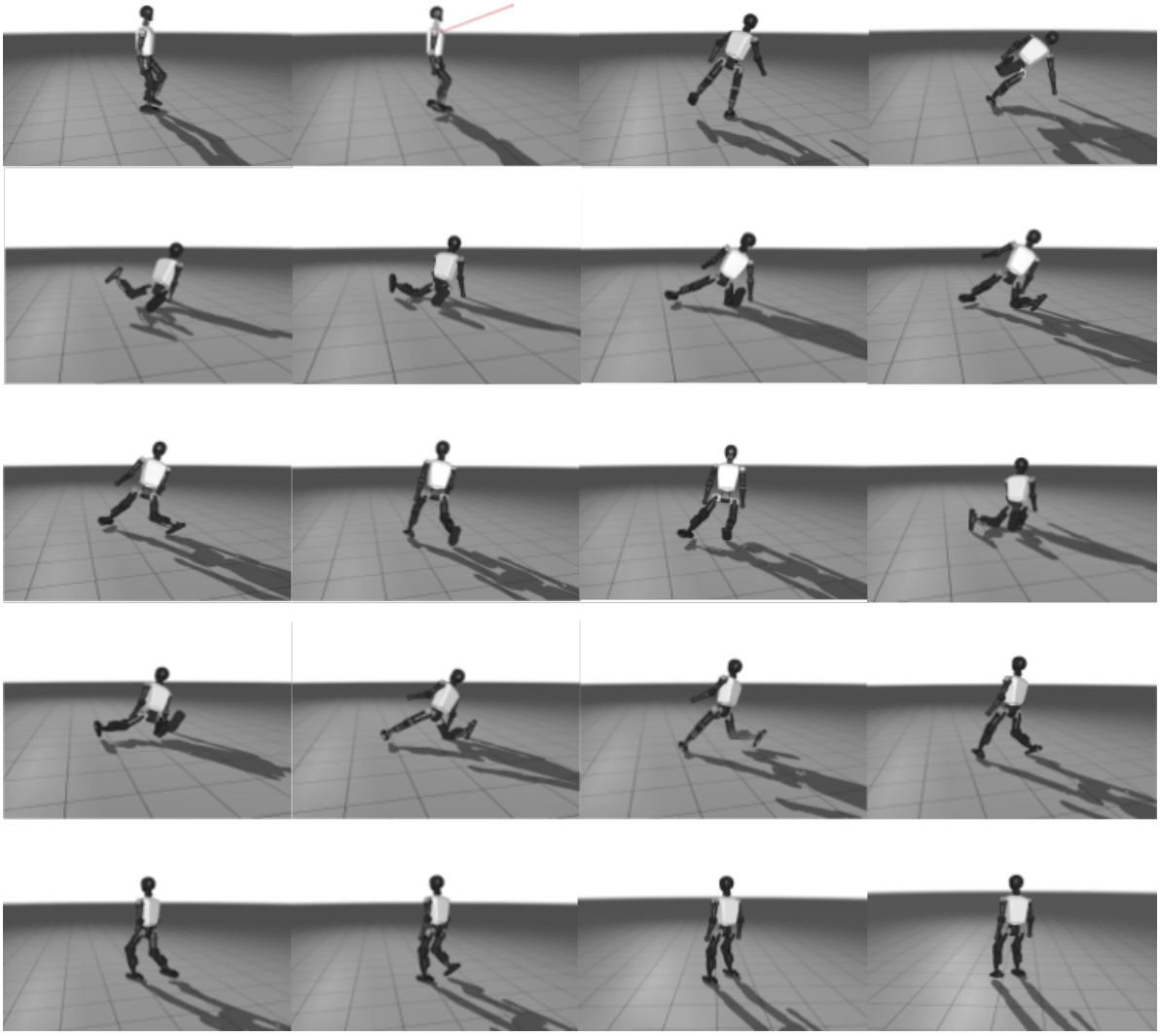

Figure C.40: Baseline forward push evaluation 12

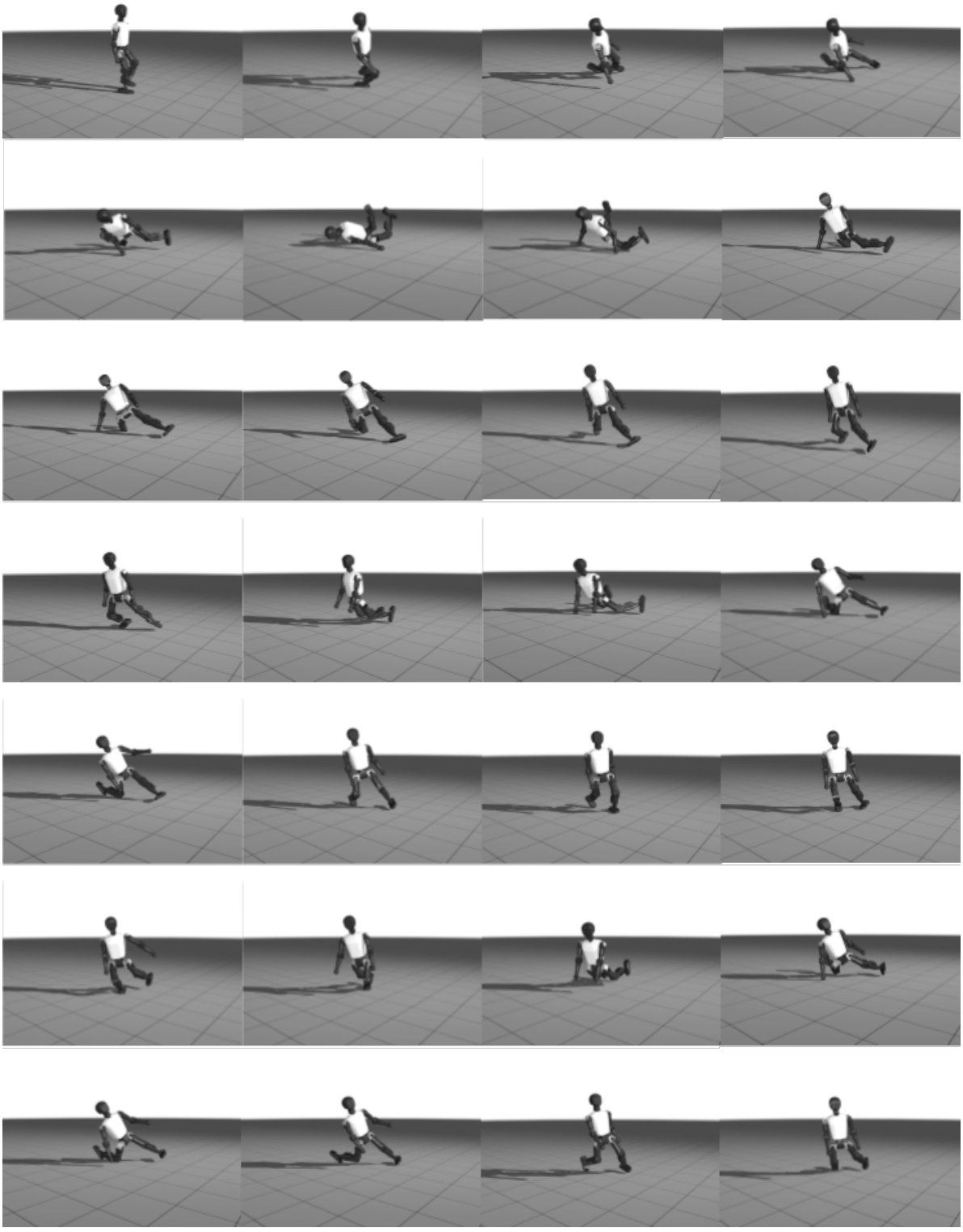

Figure C.41: (a) Baseline forward push evaluation 13

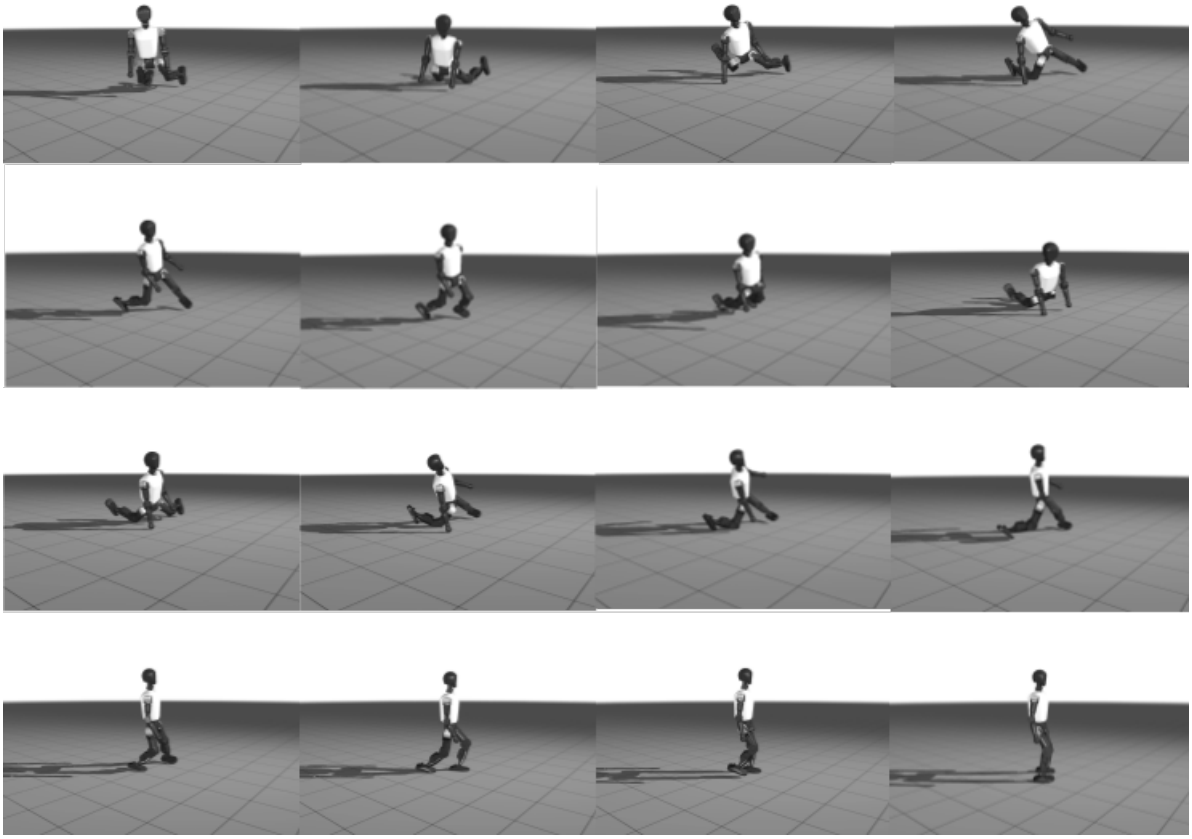

Figure C.42: (b) Baseline forward push evaluation 13

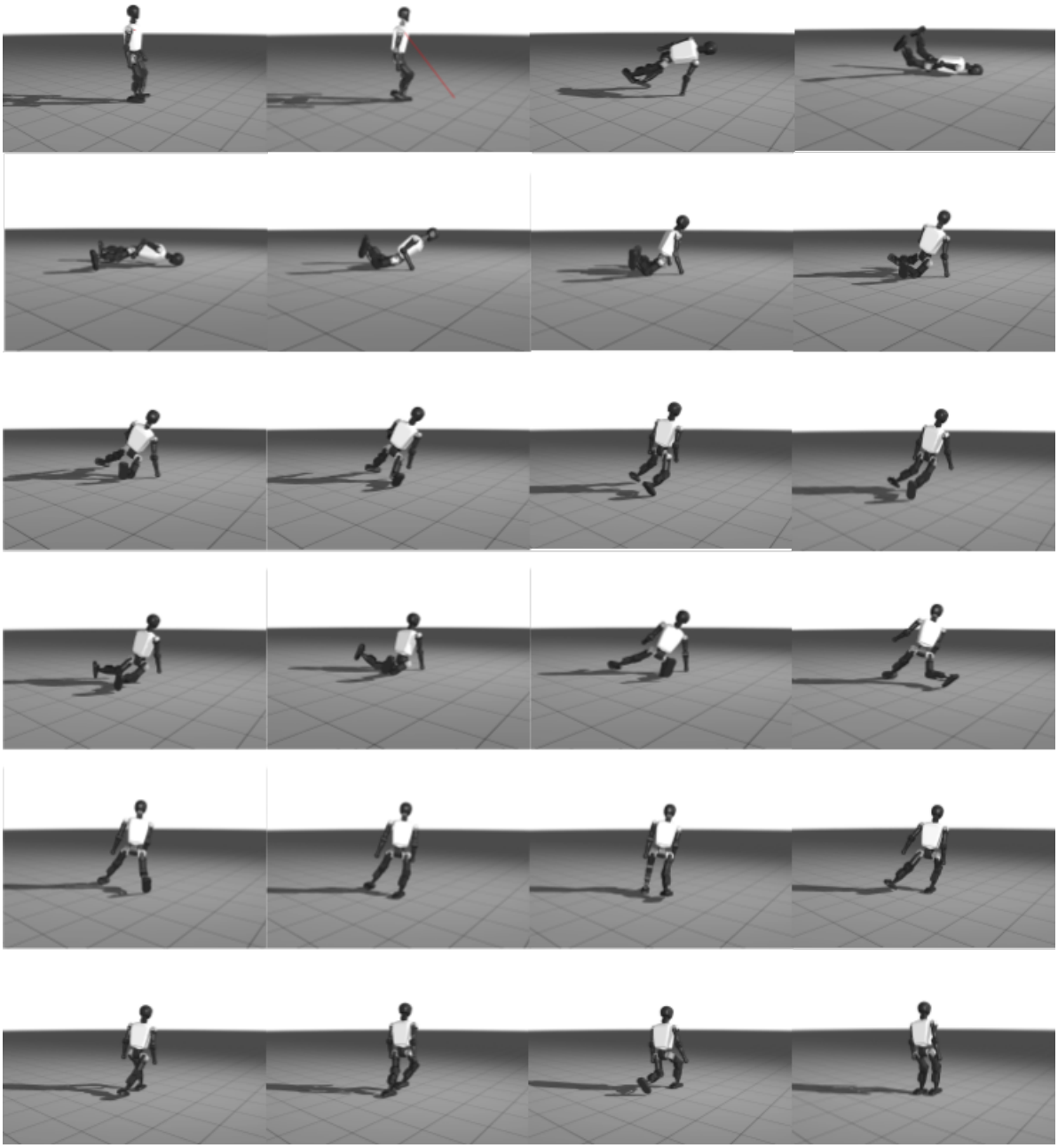

Figure C.43: Baseline forward push evaluation 14

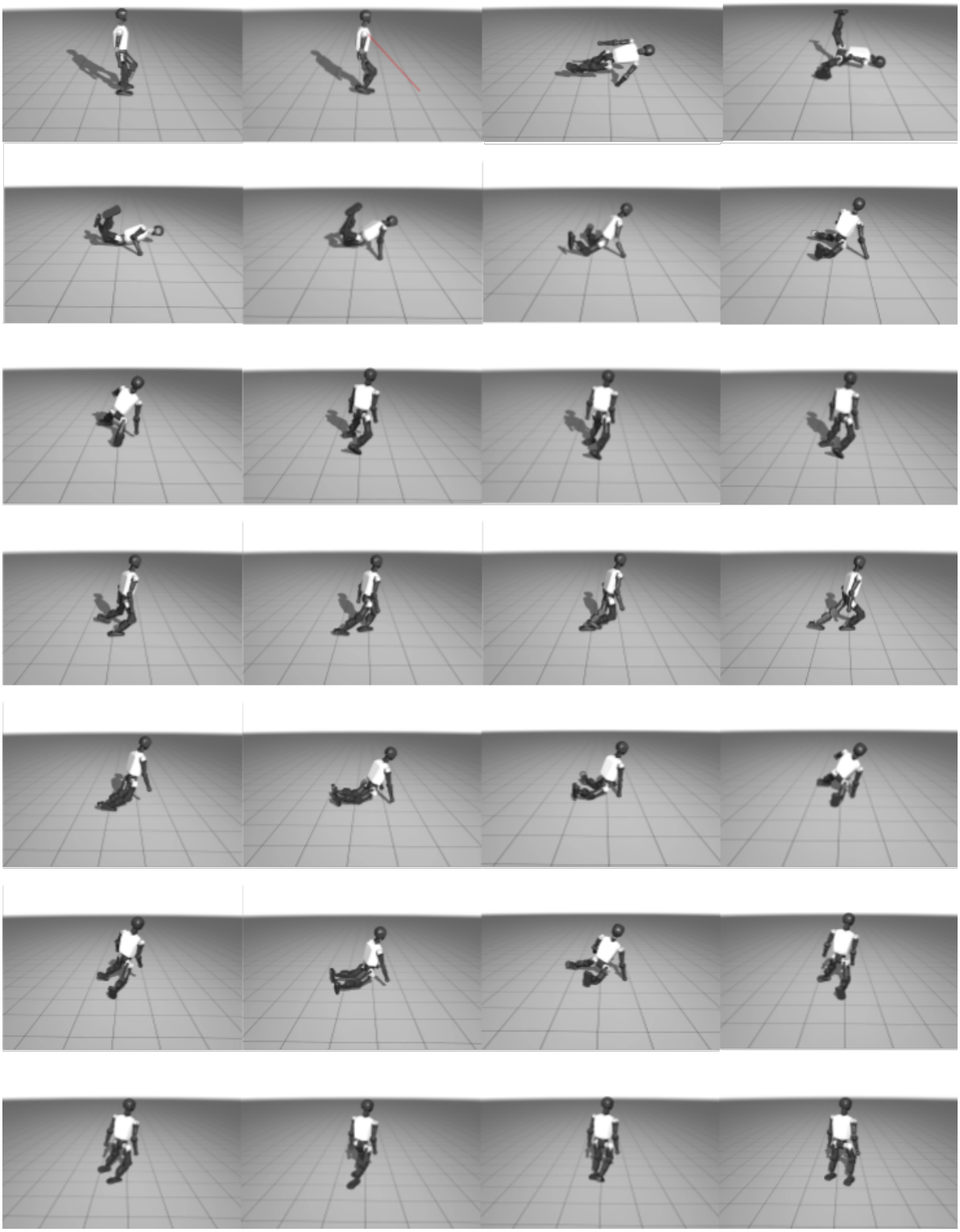

Figure C.44: Baseline forward push evaluation 15

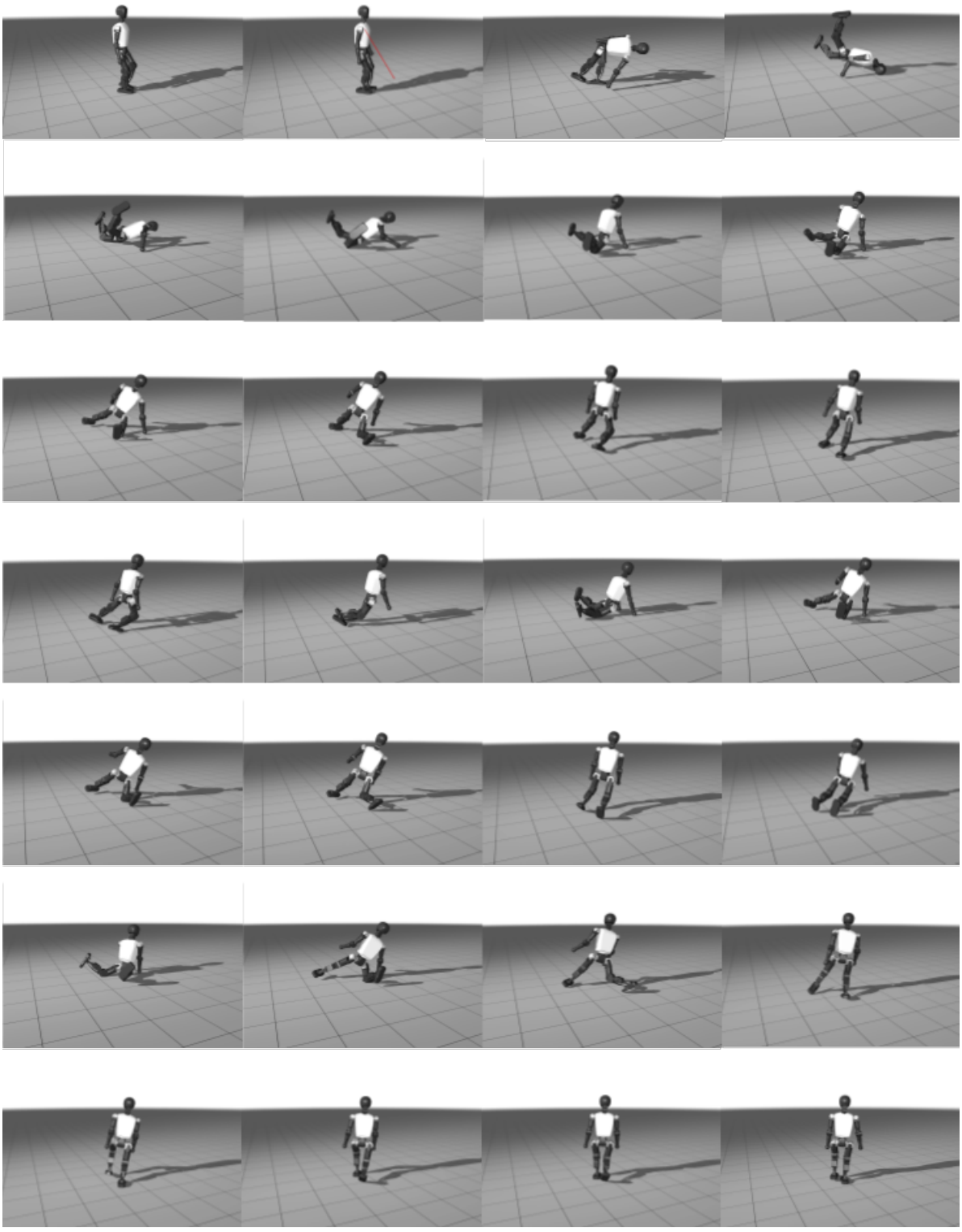

Figure C.45: Baseline forward push evaluation 16

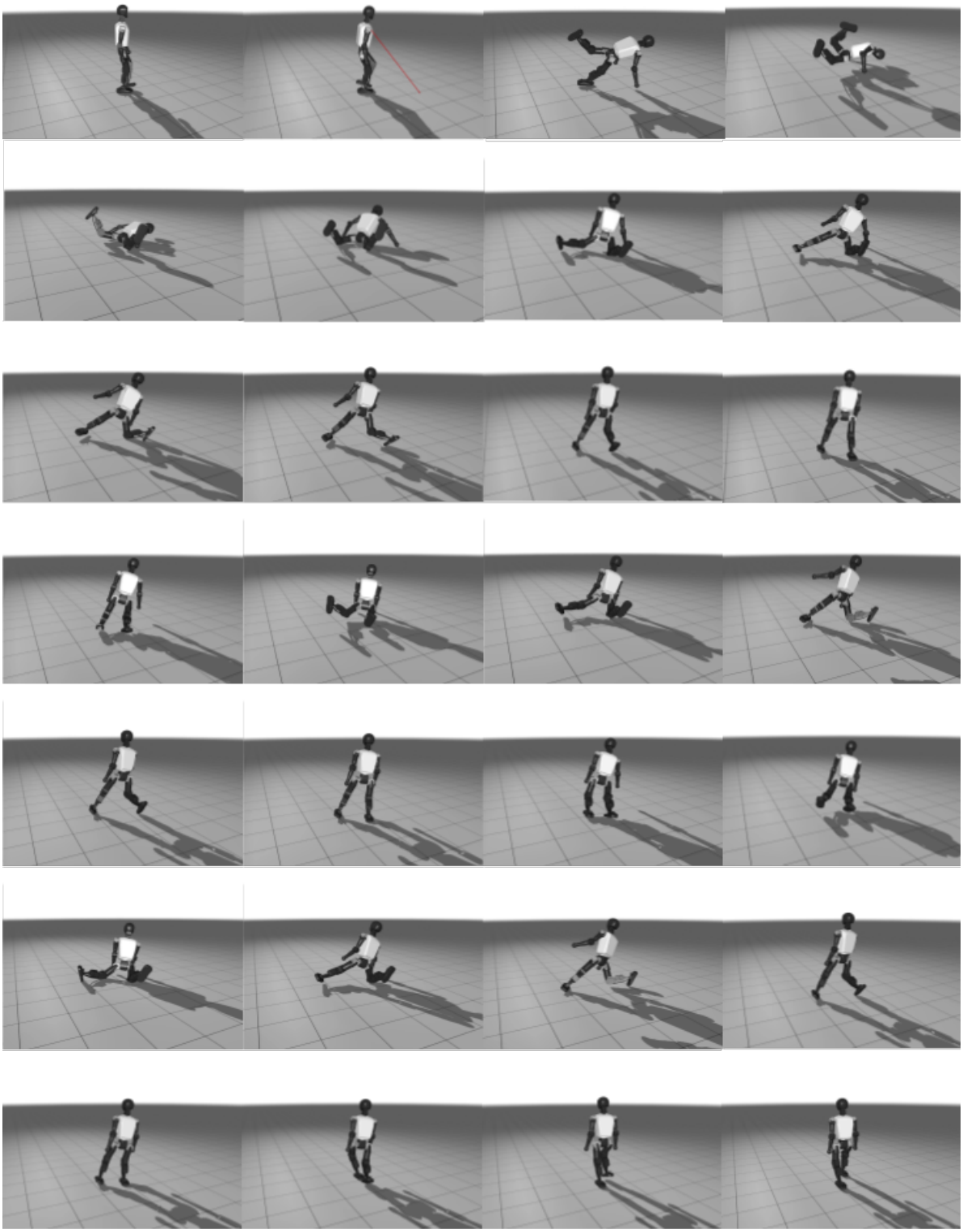

Figure C.46: Baseline forward push evaluation 17

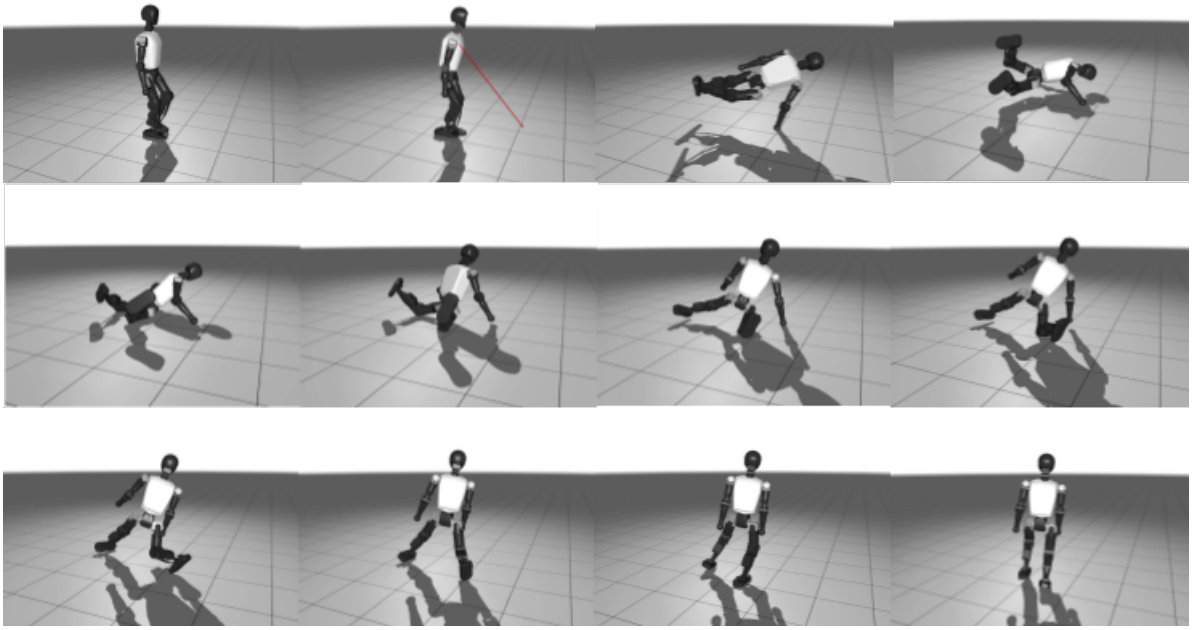

Figure C.47: Baseline forward push evaluation 18

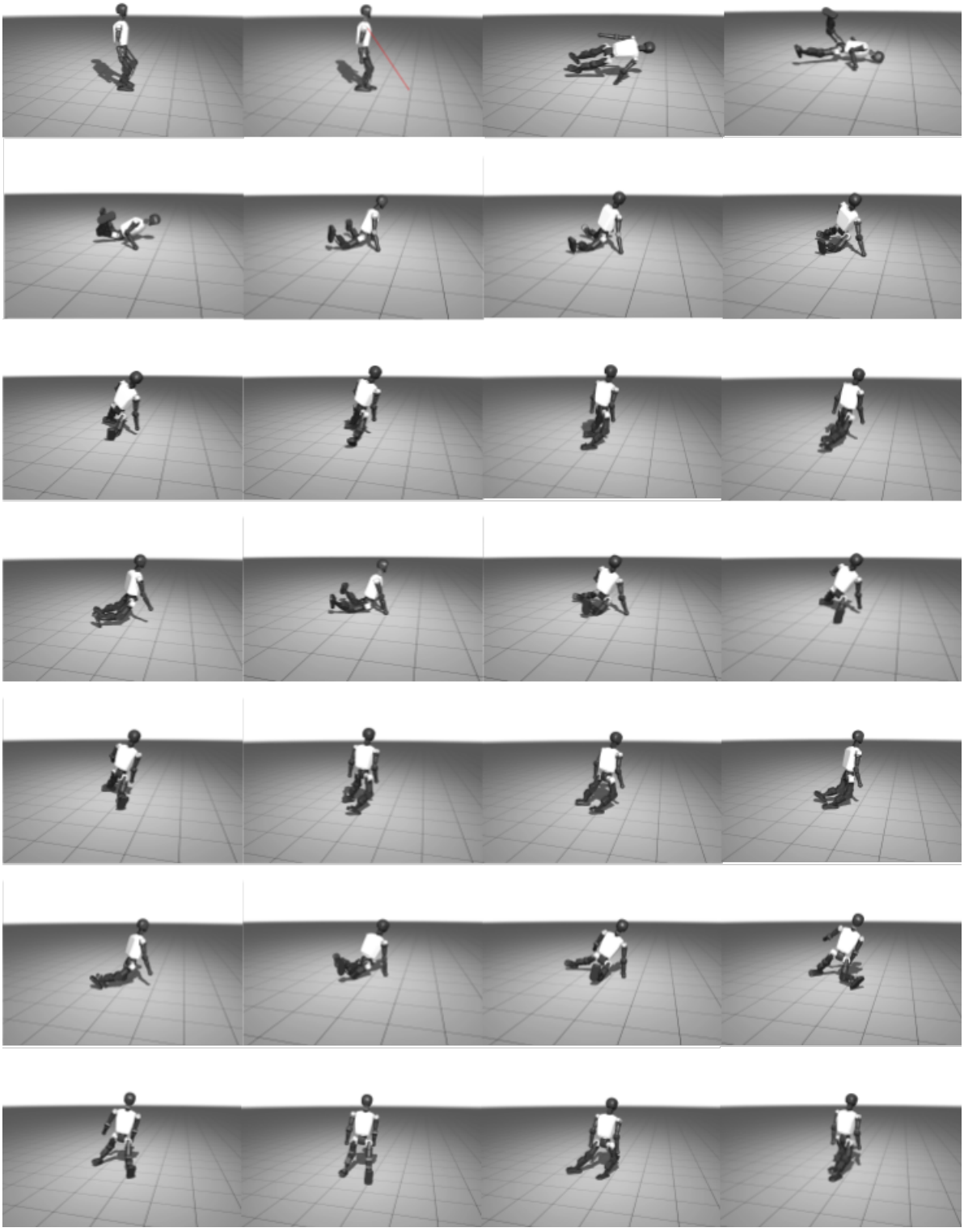

Figure C.48: (a) Baseline forward push evaluation 19

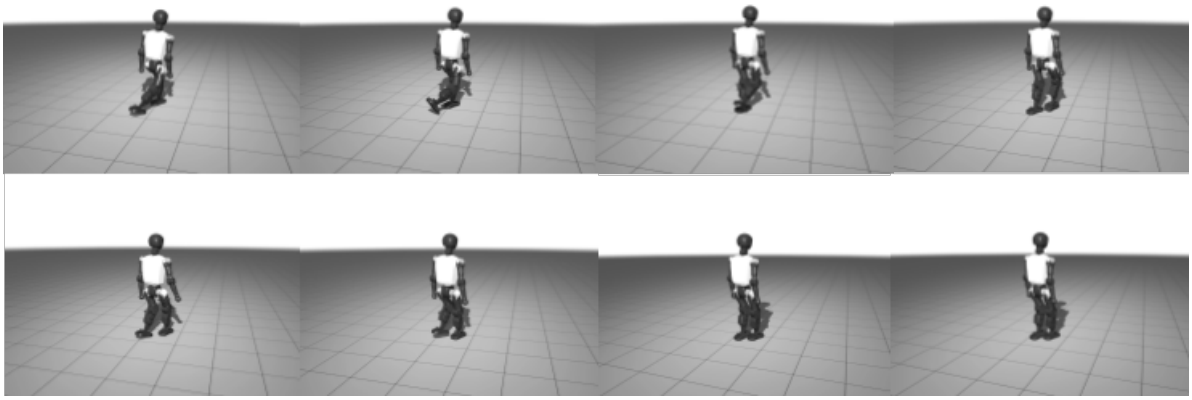

Figure C.49: (b) Baseline forward push evaluation 19

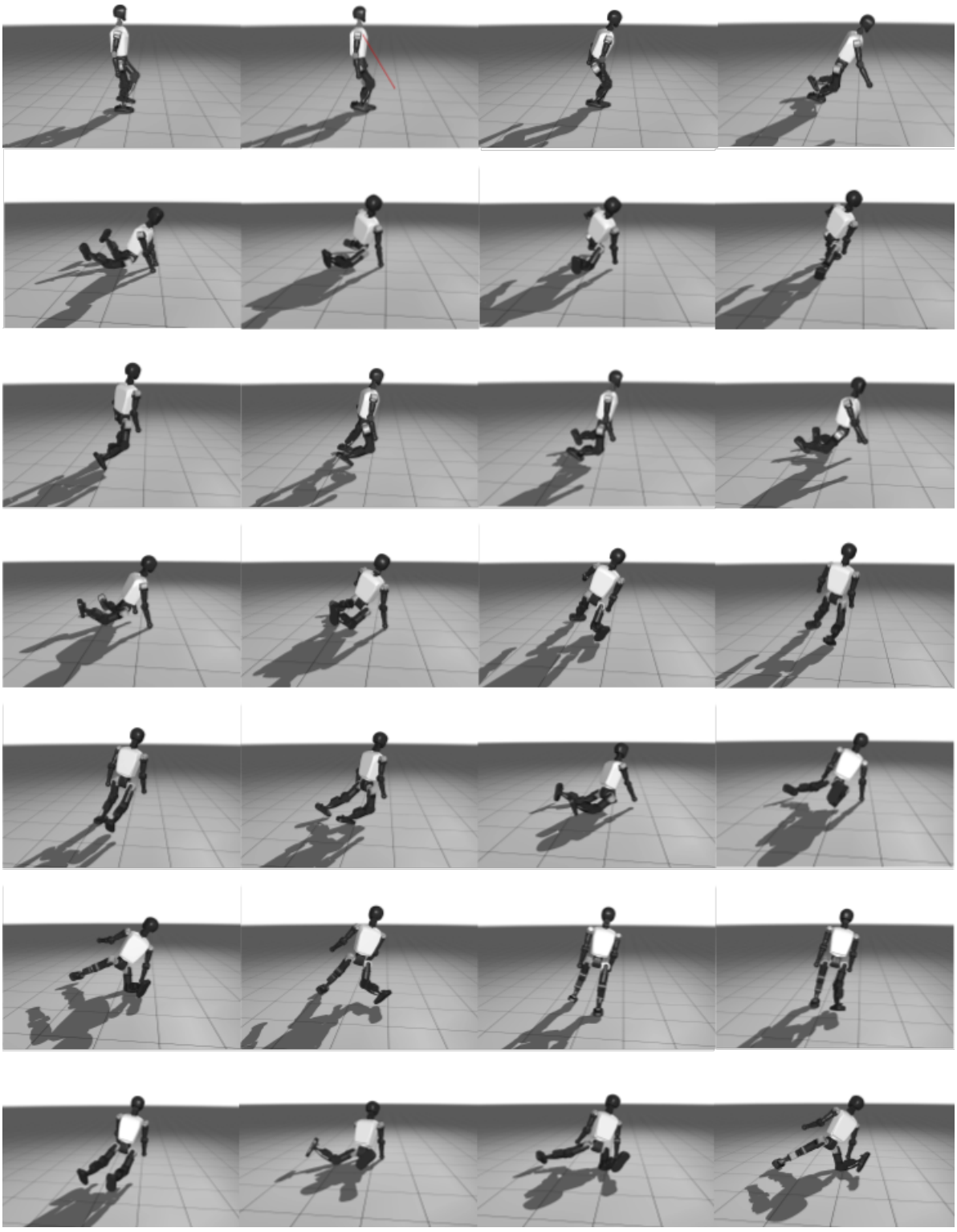

Figure C.50: (a) Baseline forward push evaluation 20

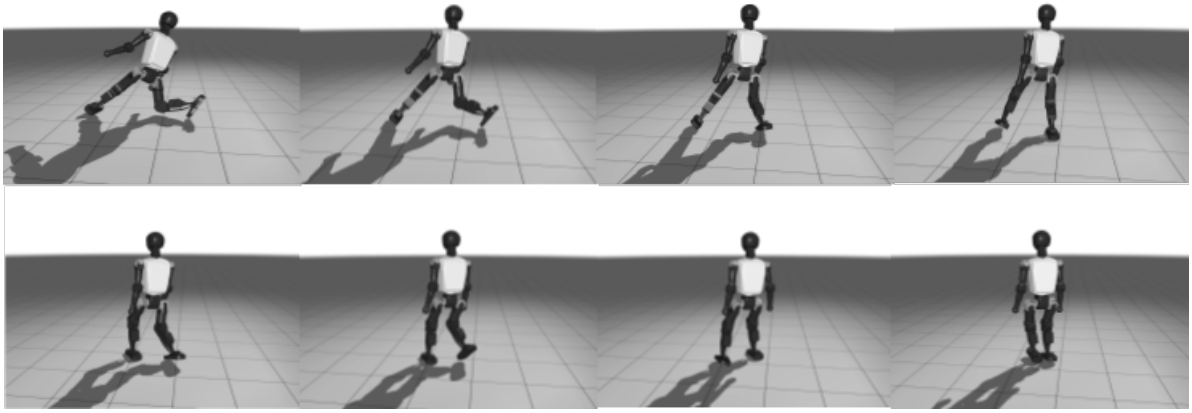

Figure C.51: (b) Baseline forward push evaluation 20

Appendix D

Backward Push Evaluation Trials

DAgger-100k backward push evaluations

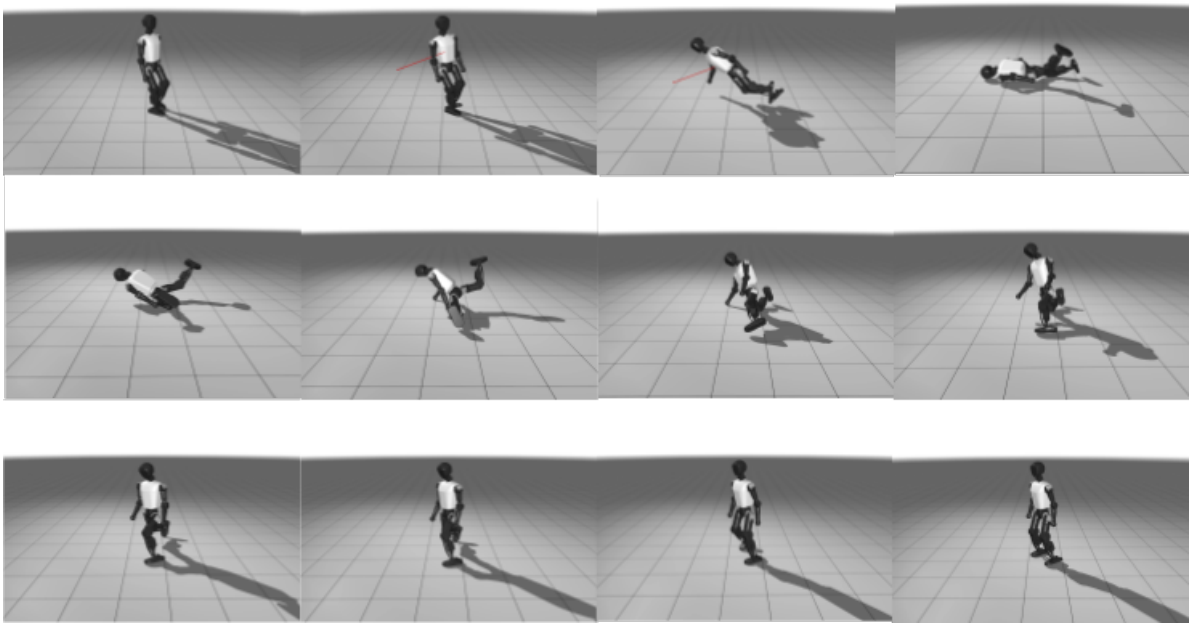

Figure D.1: DAgger-100k backwards push evaluation 1

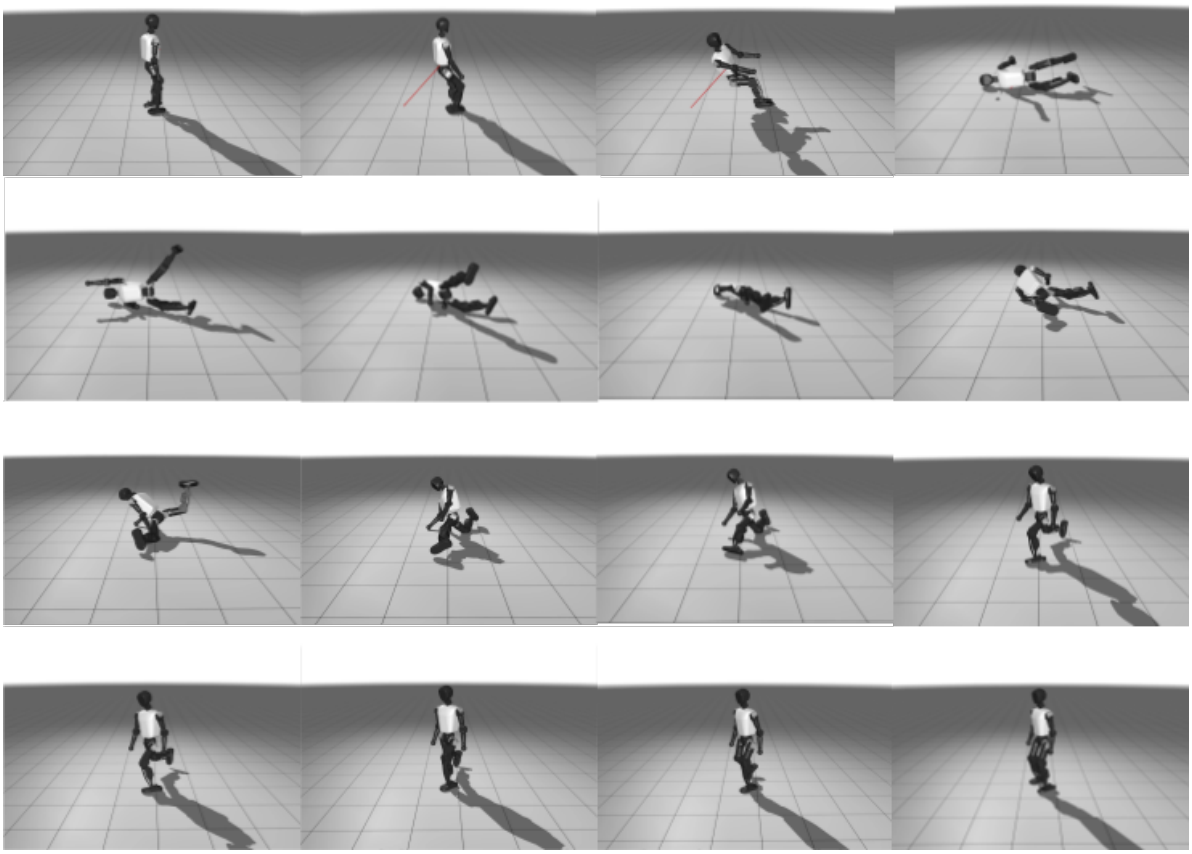

Figure D.2: DAgger-100k backwards push evaluation 2

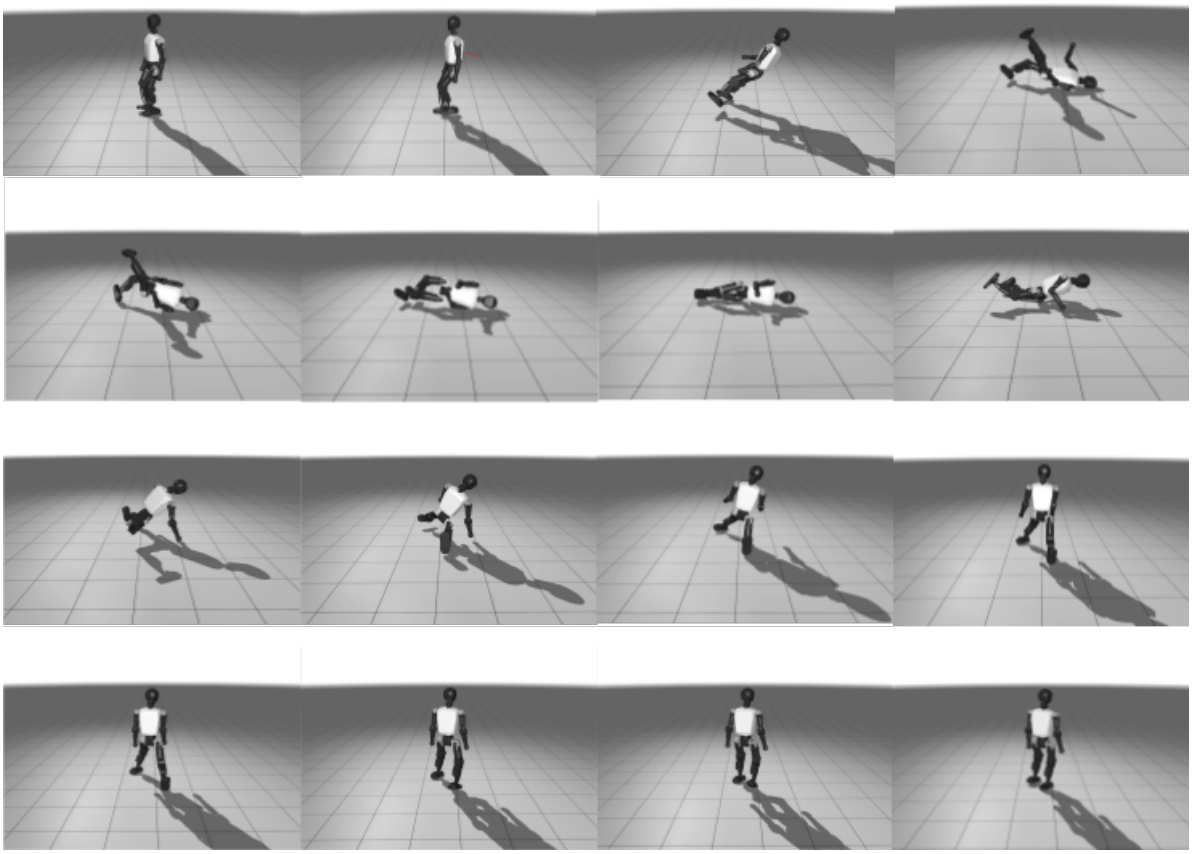

Figure D.3: DAgger-100k backwards push evaluation 3

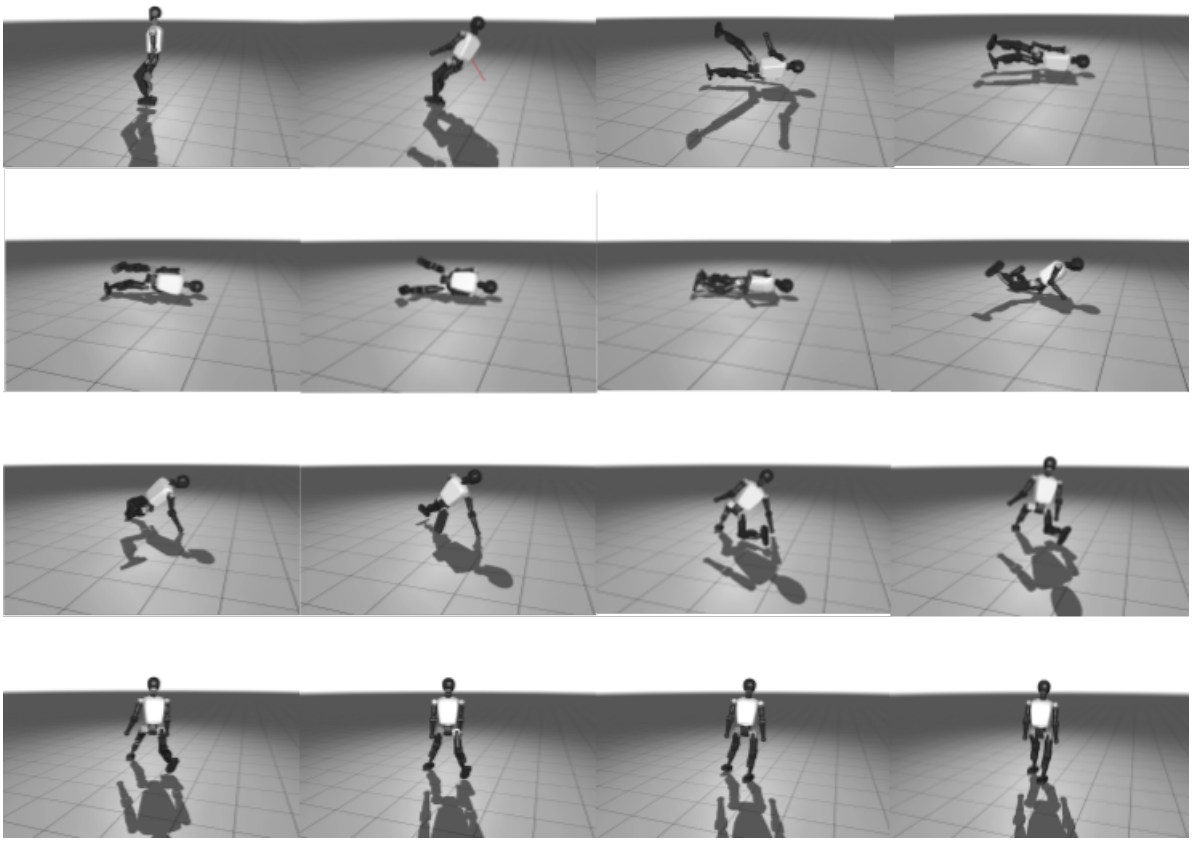

Figure D.4: DAgger-100k backwards push evaluation 4

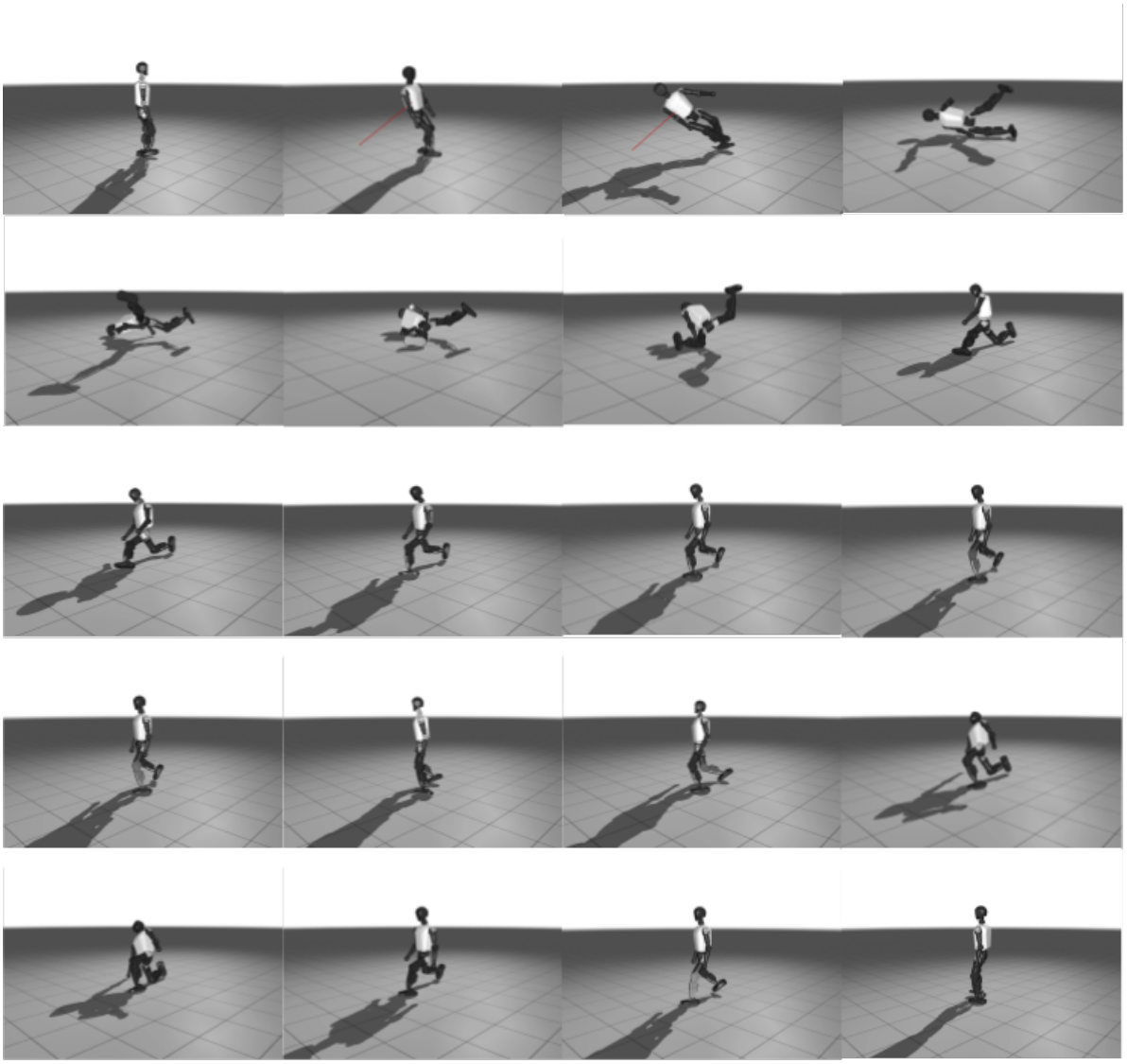

Figure D.5: DAgger-100k backwards push evaluation 5

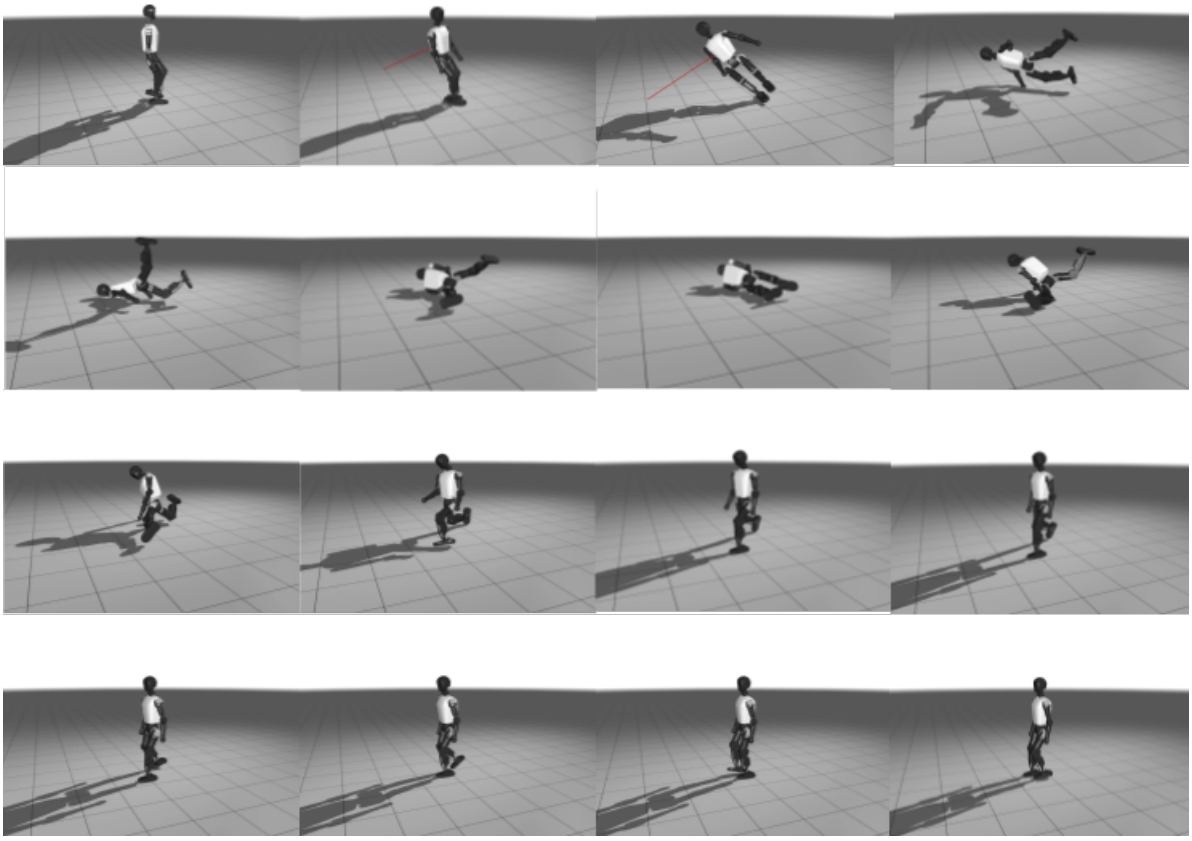

Figure D.6: DAgger-100k backwards push evaluation 6

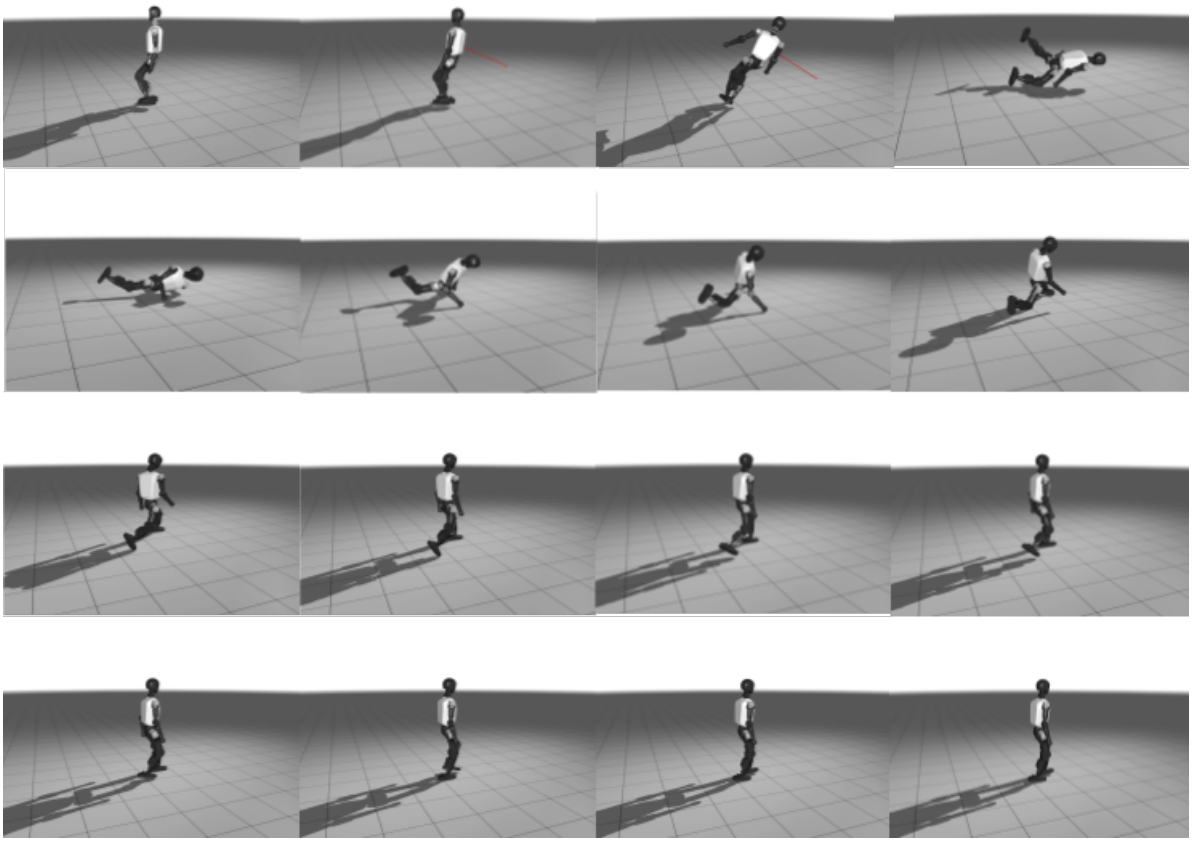

Figure D.7: DAgger-100k backwards push evaluation 7

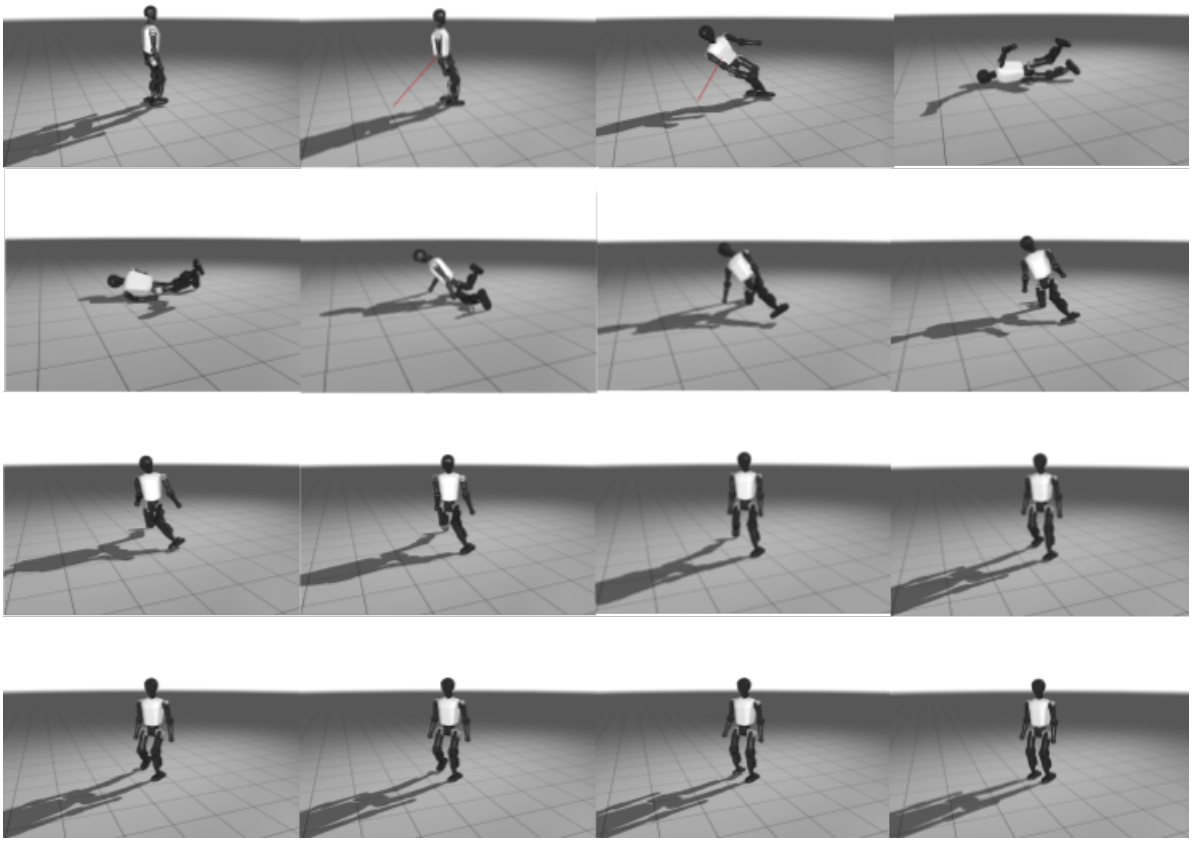

Figure D.8: DAgger-100k backwards push evaluation 8

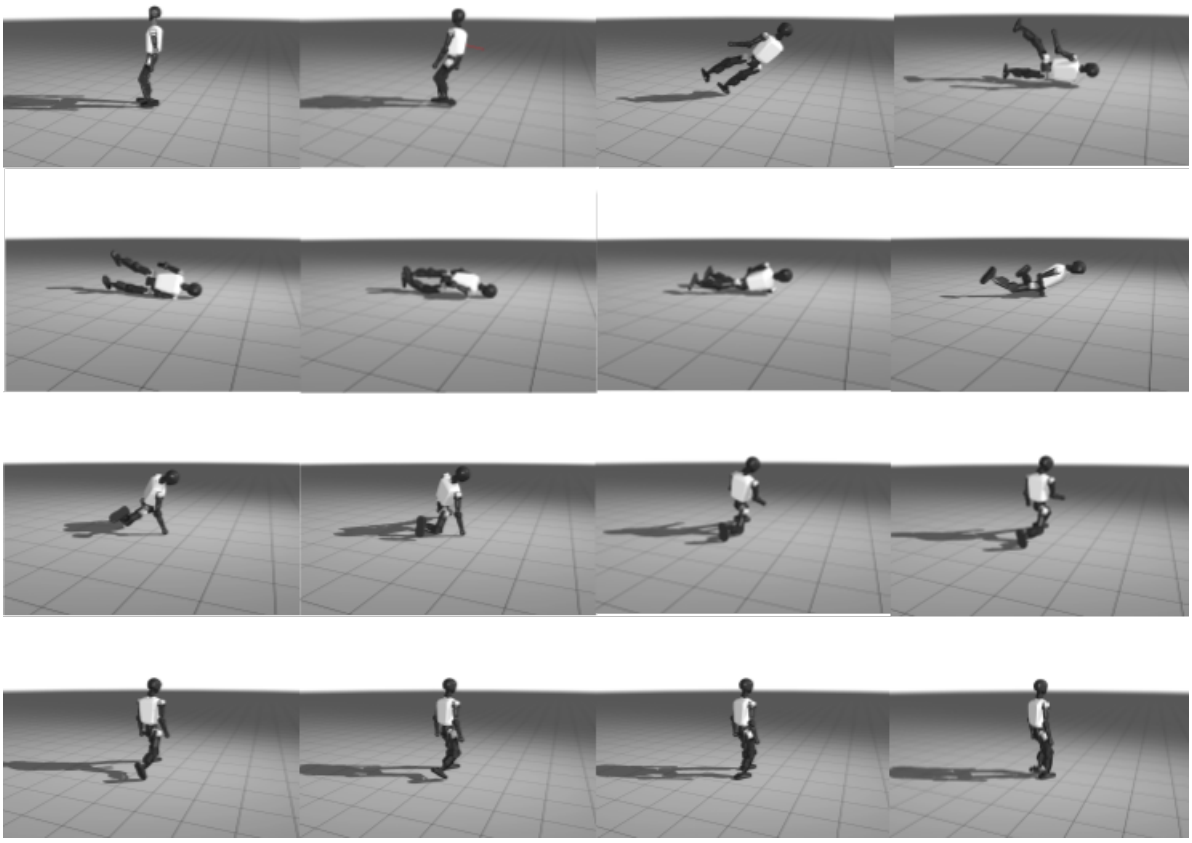

Figure D.9: DAgger-100k backwards push evaluation 9

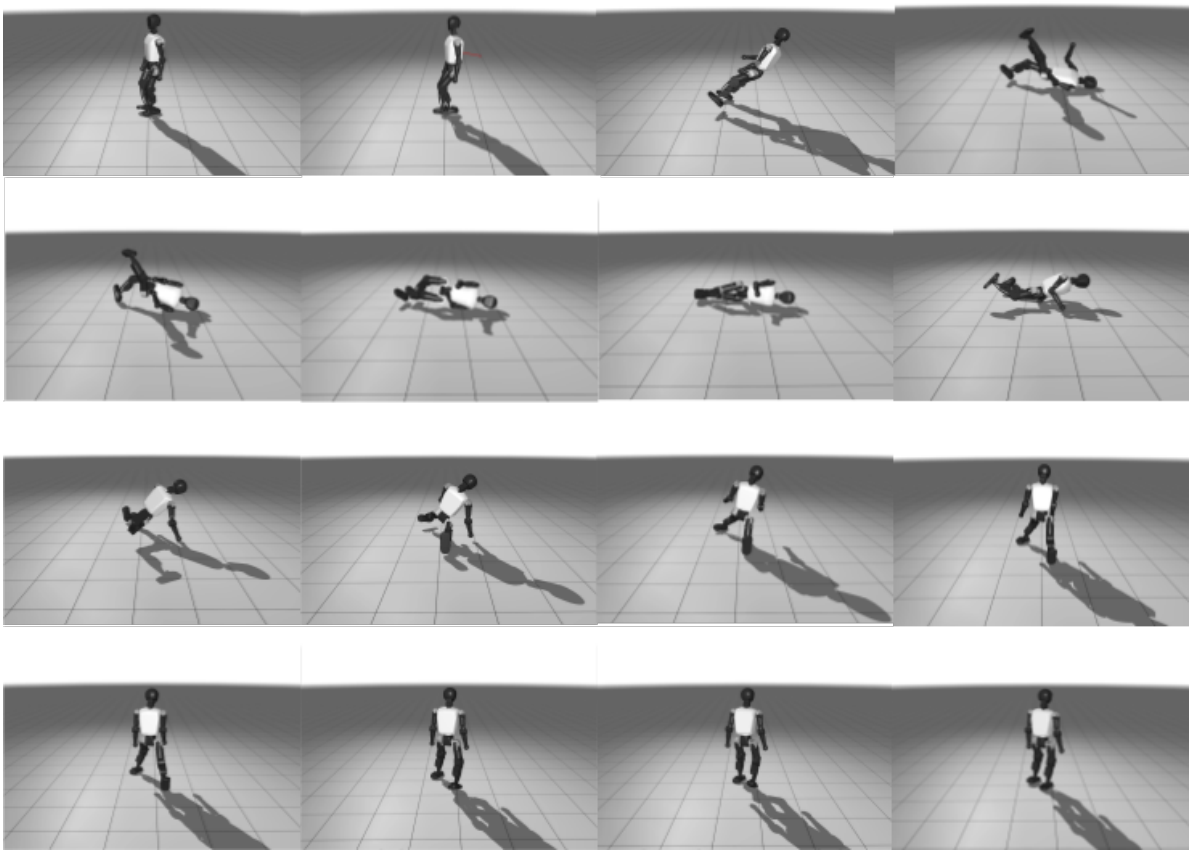

Figure D.10: DAgger-100k backwards push evaluation 10

Baseline backward push evaluations

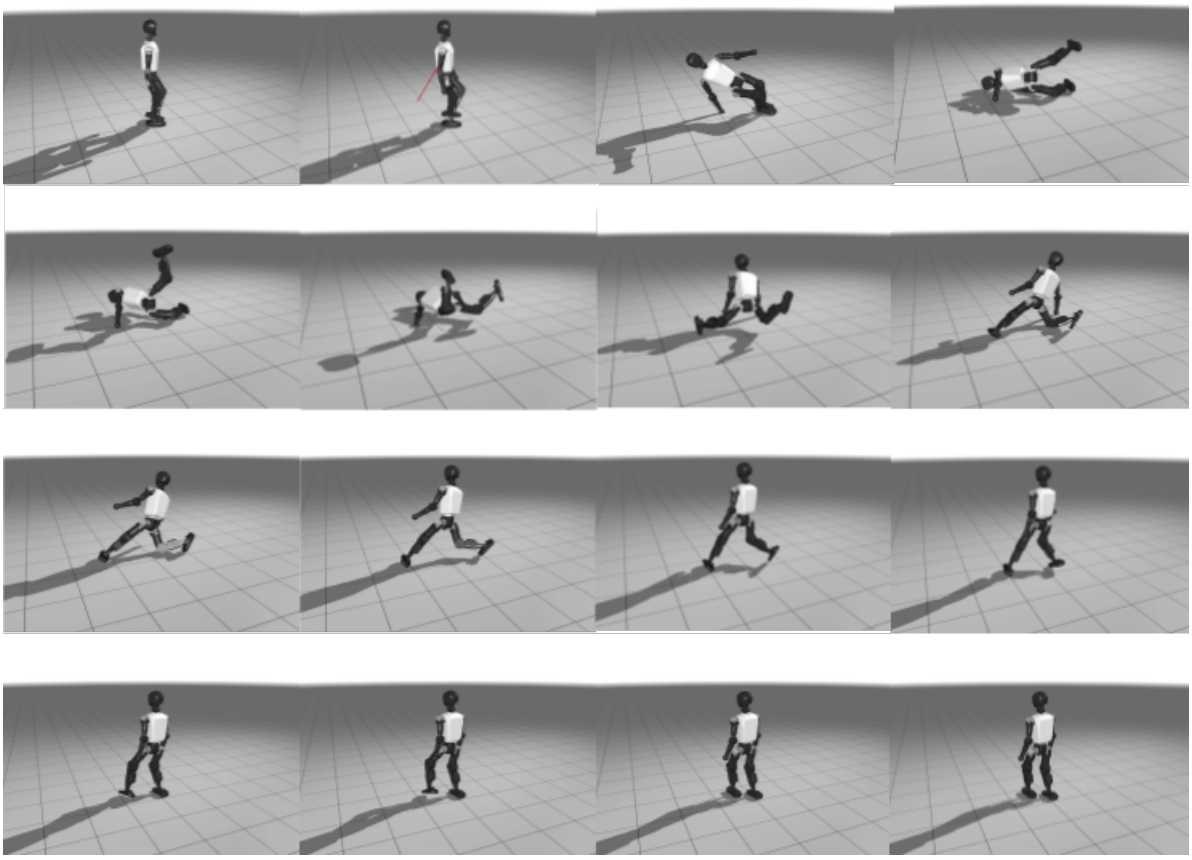

Figure D.11: Baseline backwards push evaluation 1

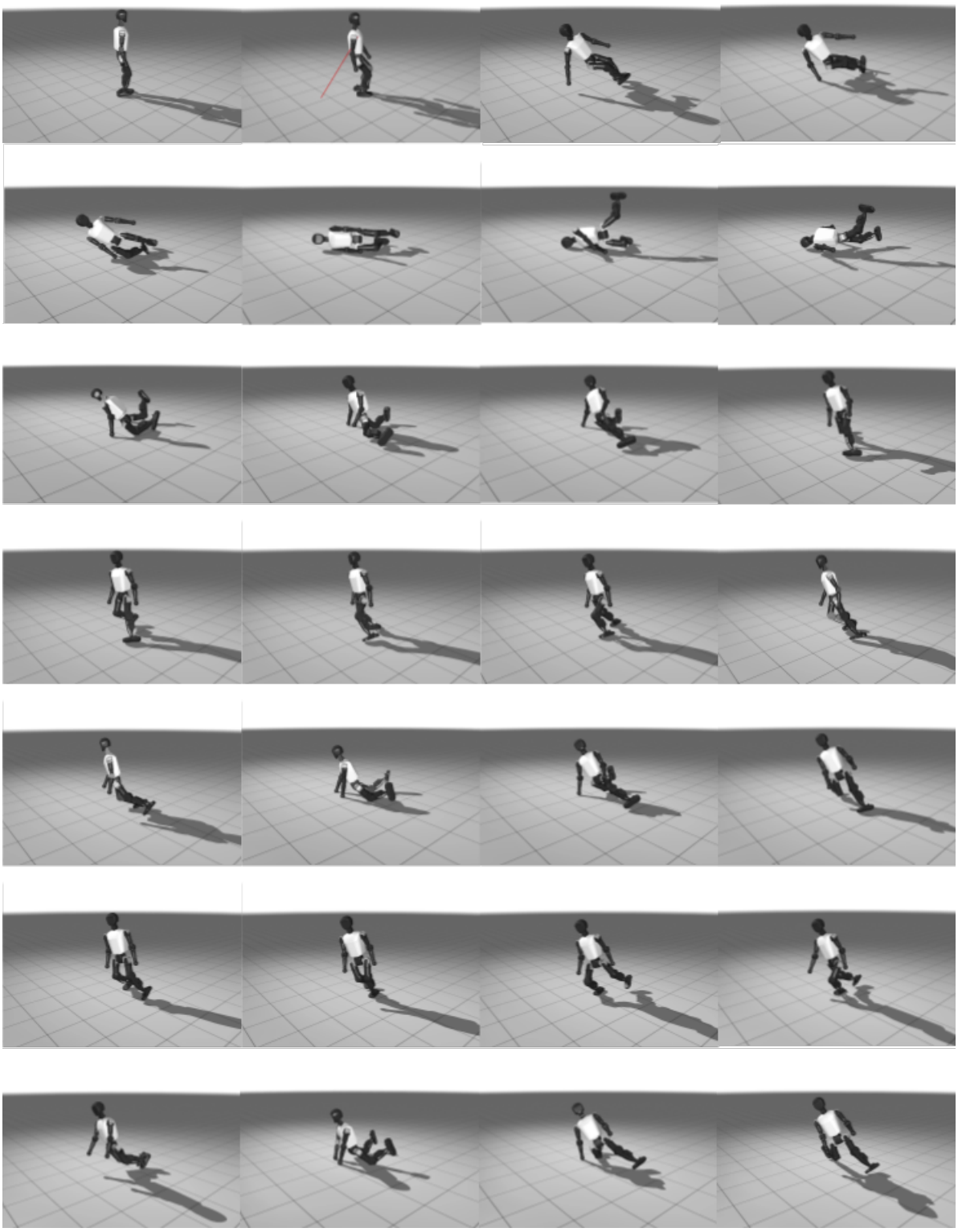

Figure D.12: (a) Baseline backwards push evaluation 2

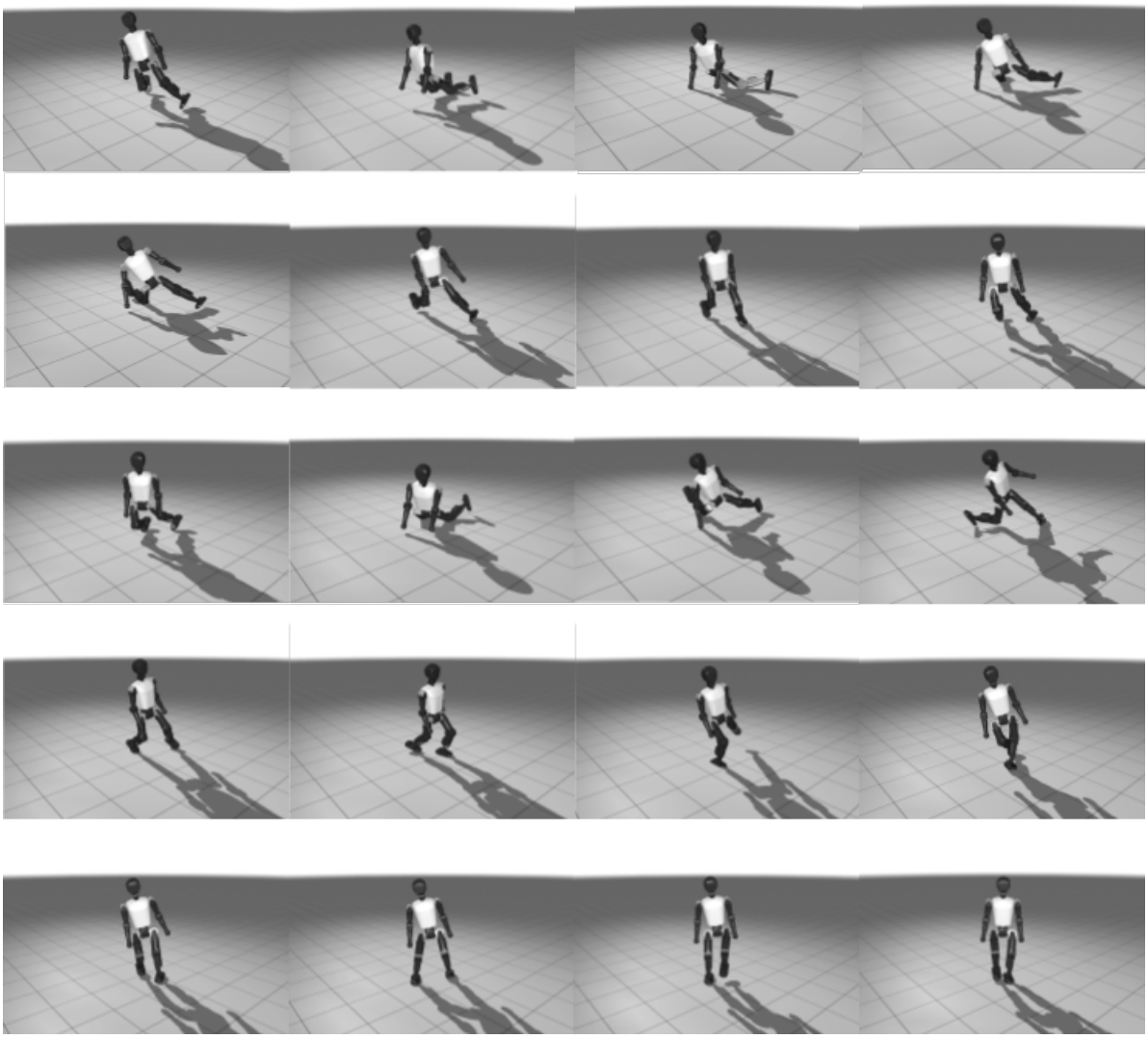

Figure D.13: (b) Baseline backwards push evaluation 2

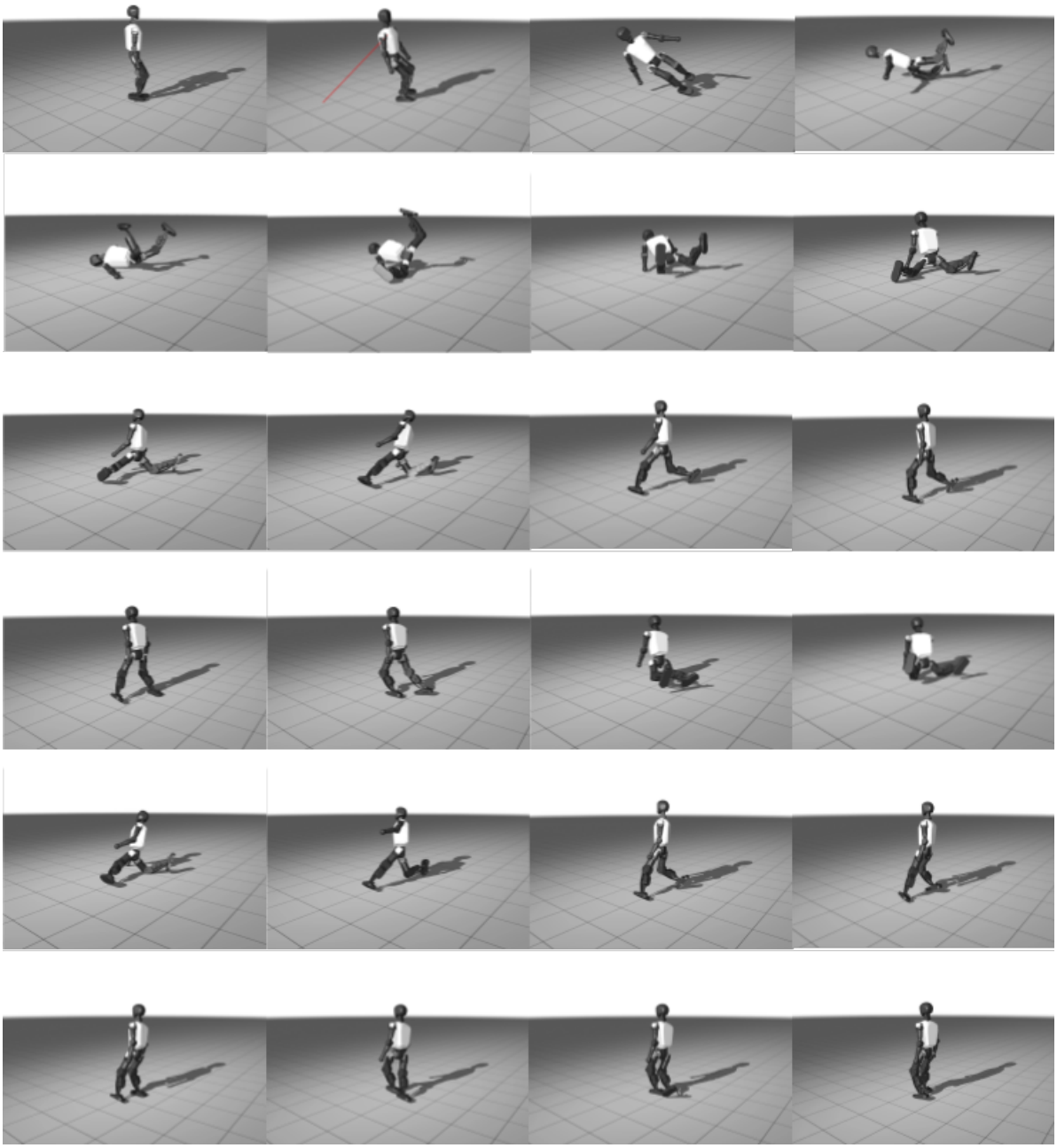

Figure D.14: Baseline backwards push evaluation 3

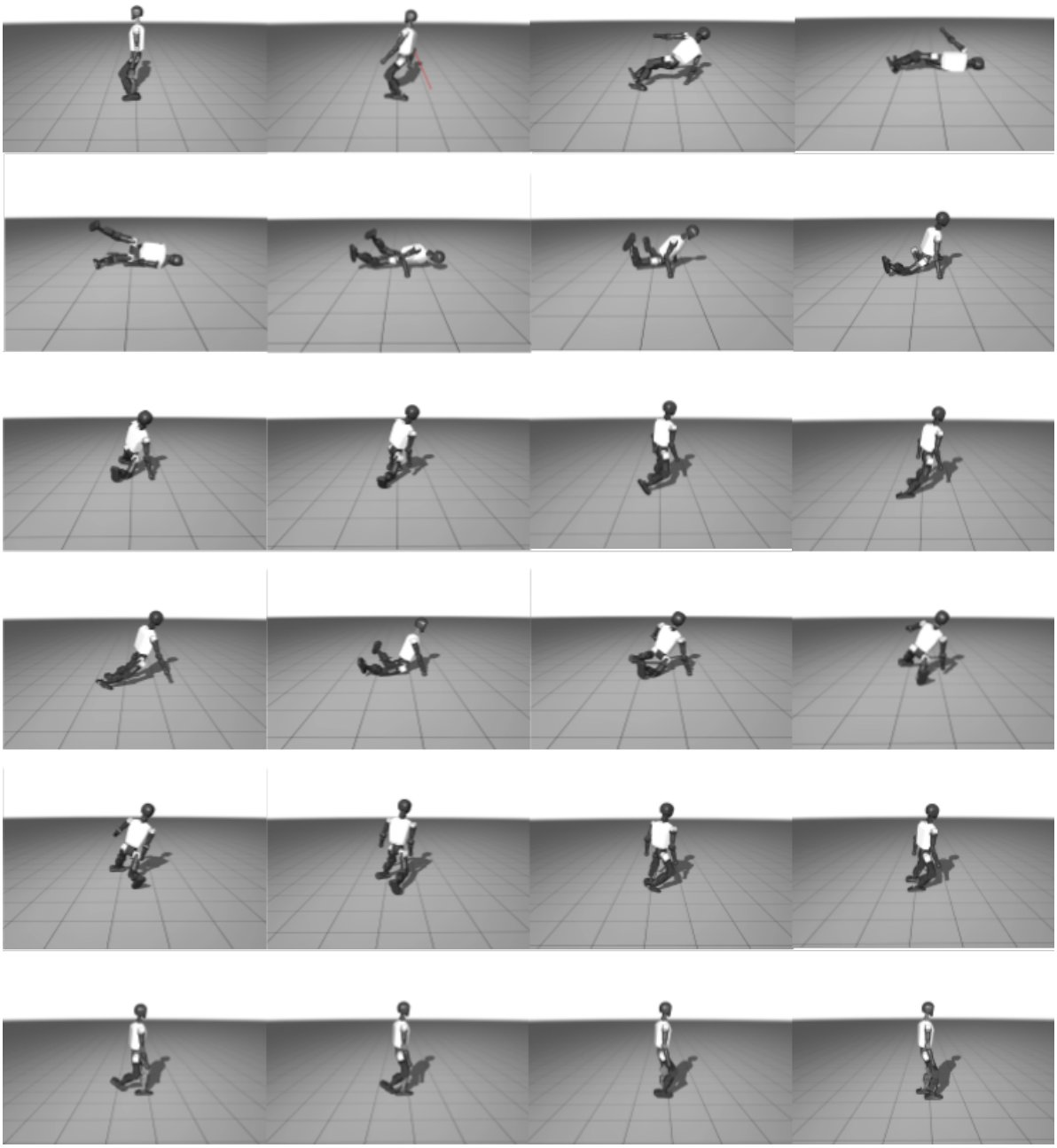

Figure D.15: Baseline backwards push evaluation 4

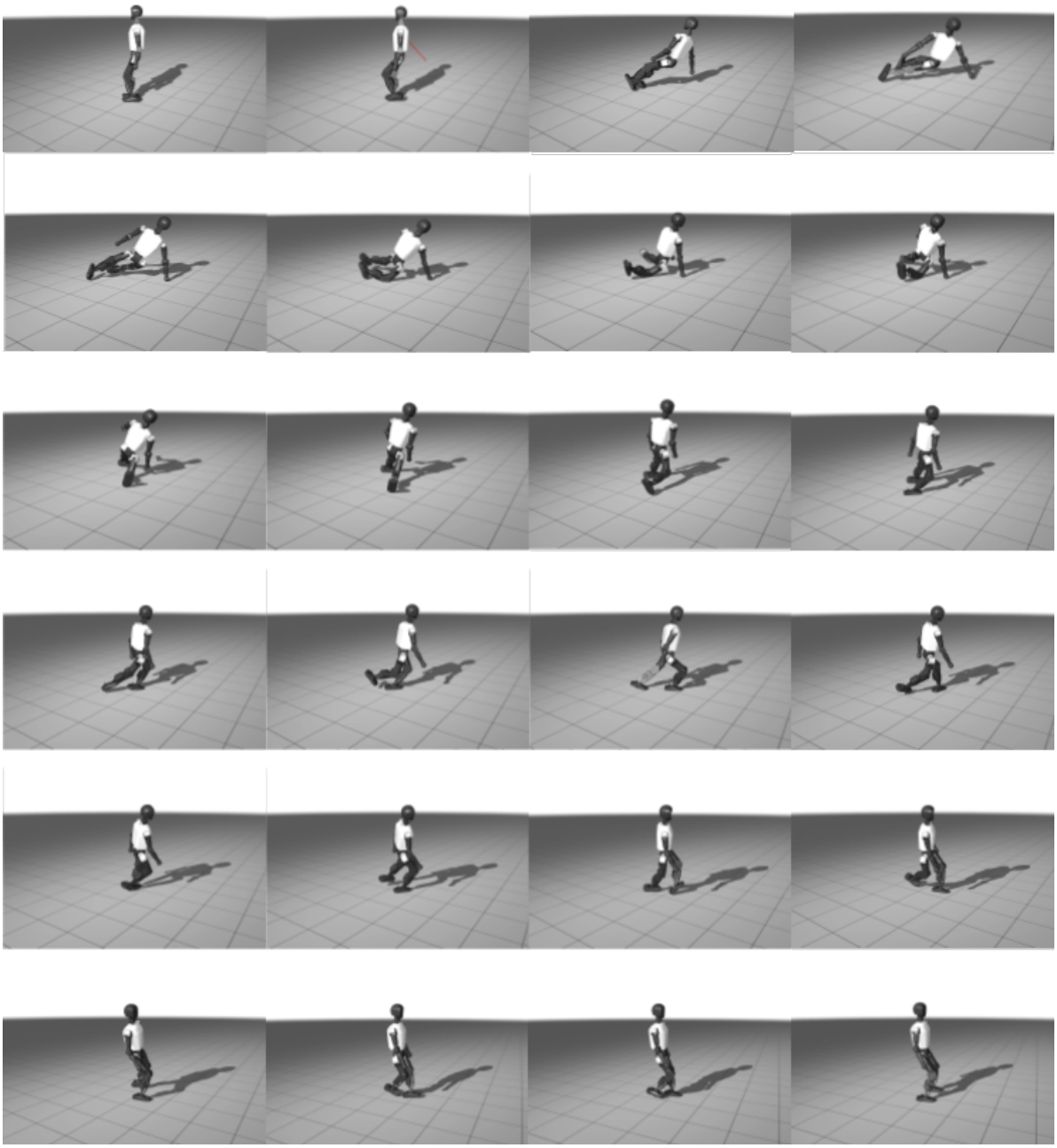

Figure D.16: Baseline backwards push evaluation 5

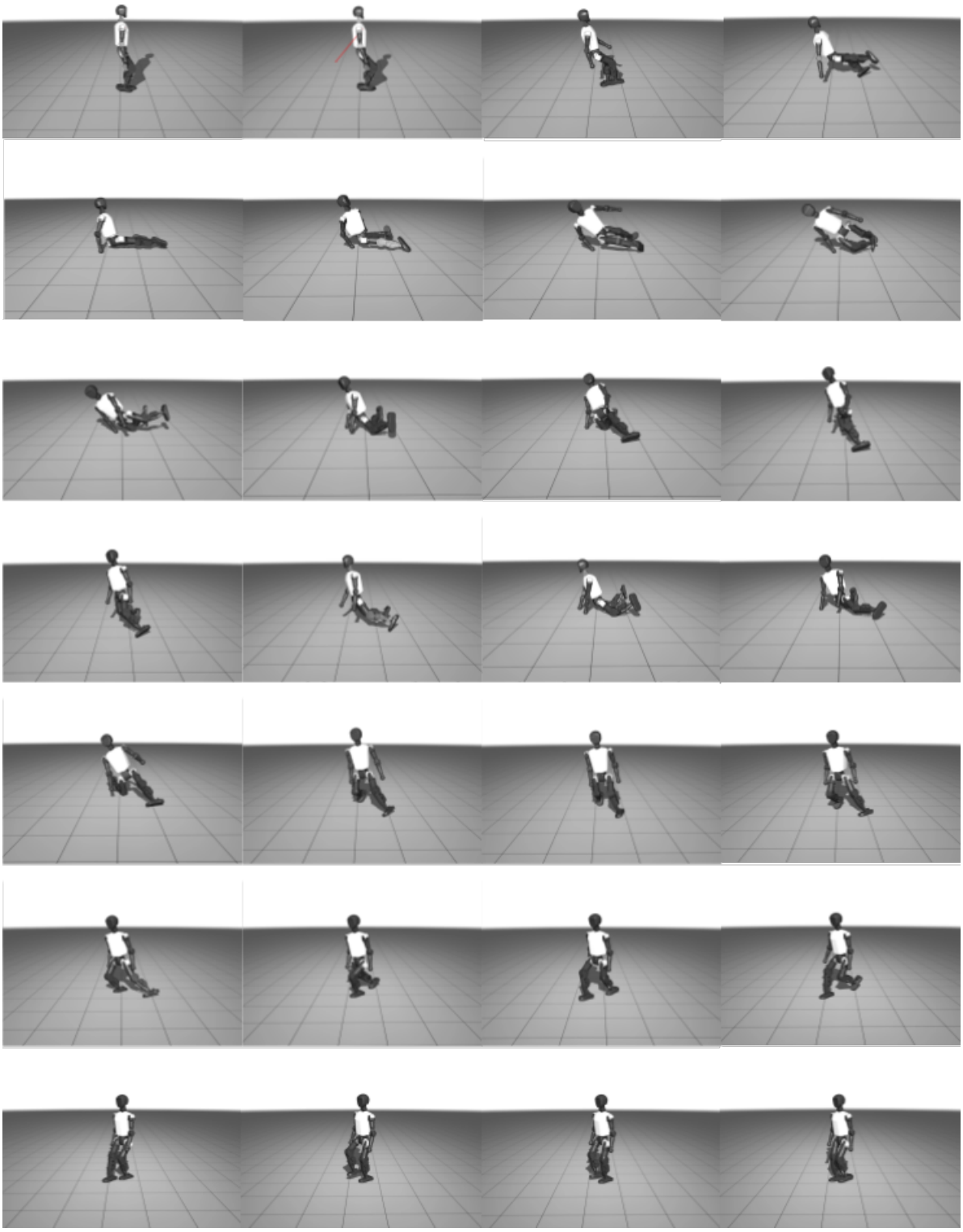

Figure D.17: Baseline backwards push evaluation 6

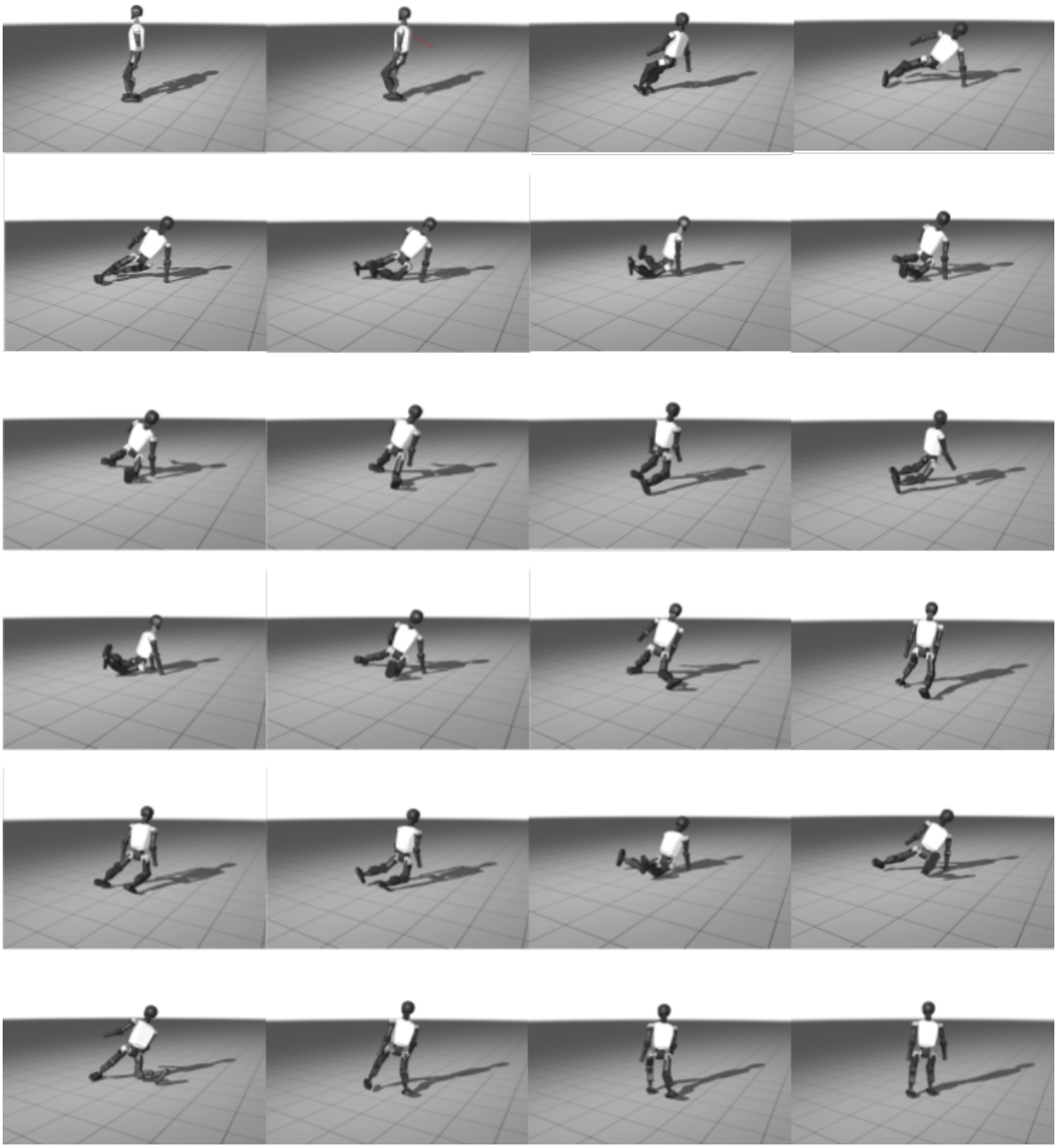

Figure D.18: Baseline backwards push evaluation 7

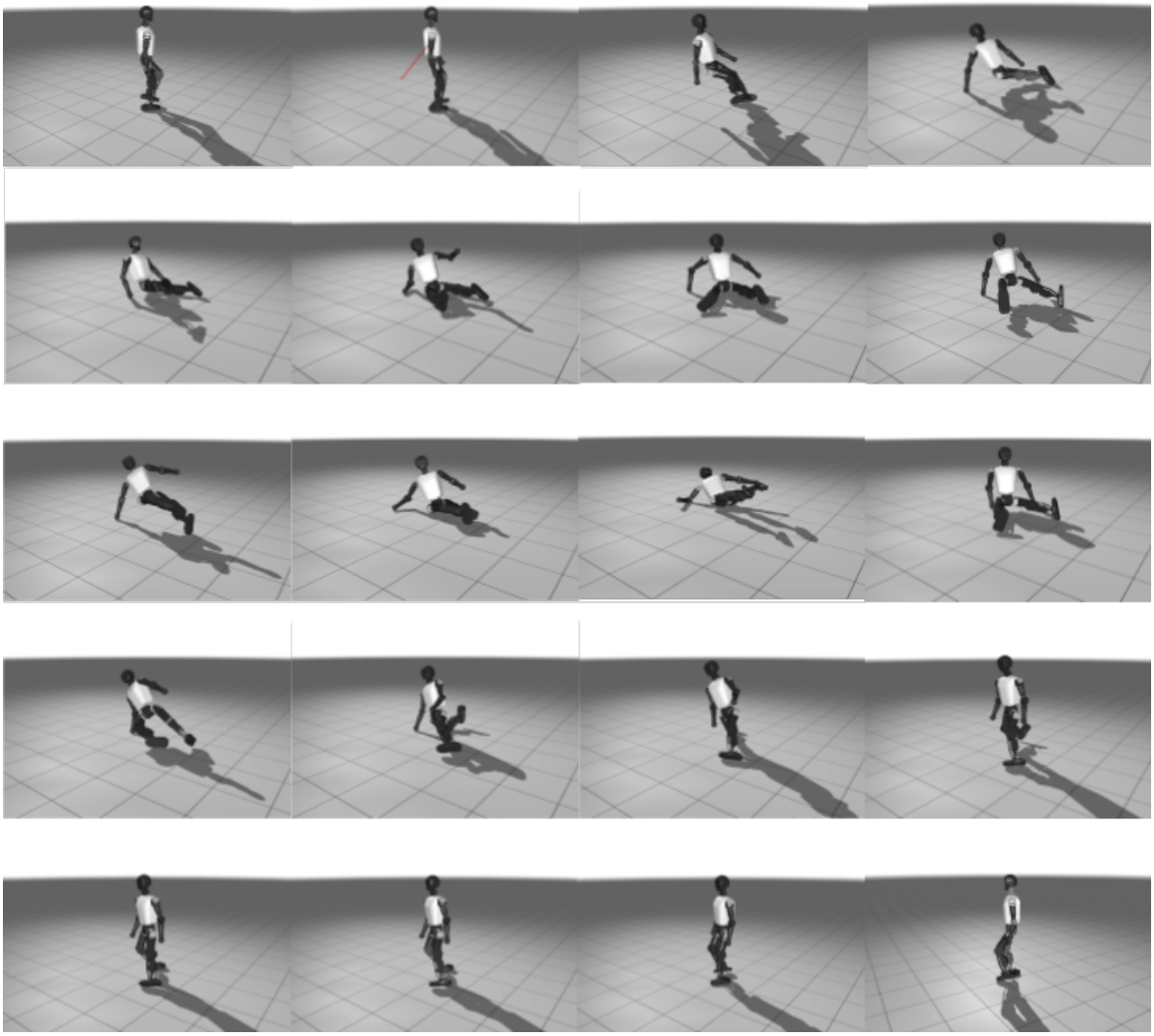

Figure D.19: Baseline backwards push evaluation 8

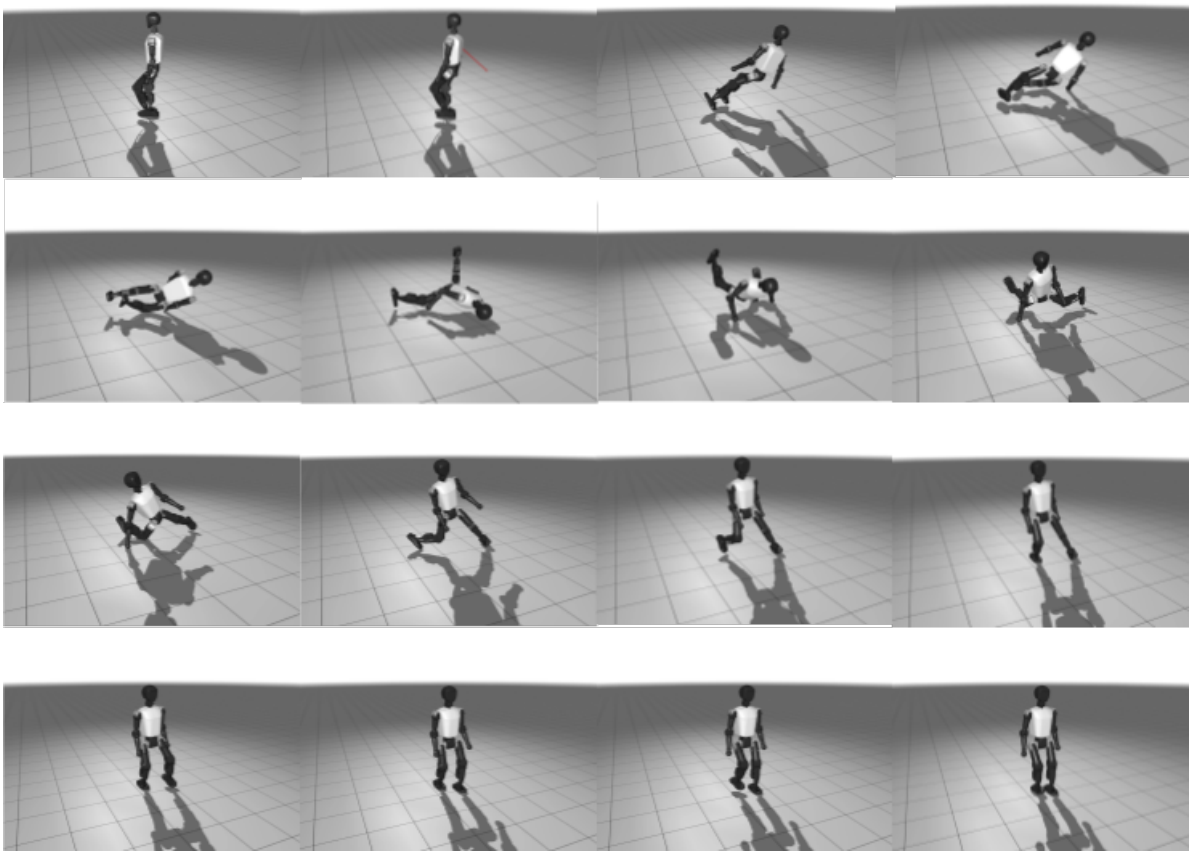

Figure D.20: Baseline backwards push evaluation 9

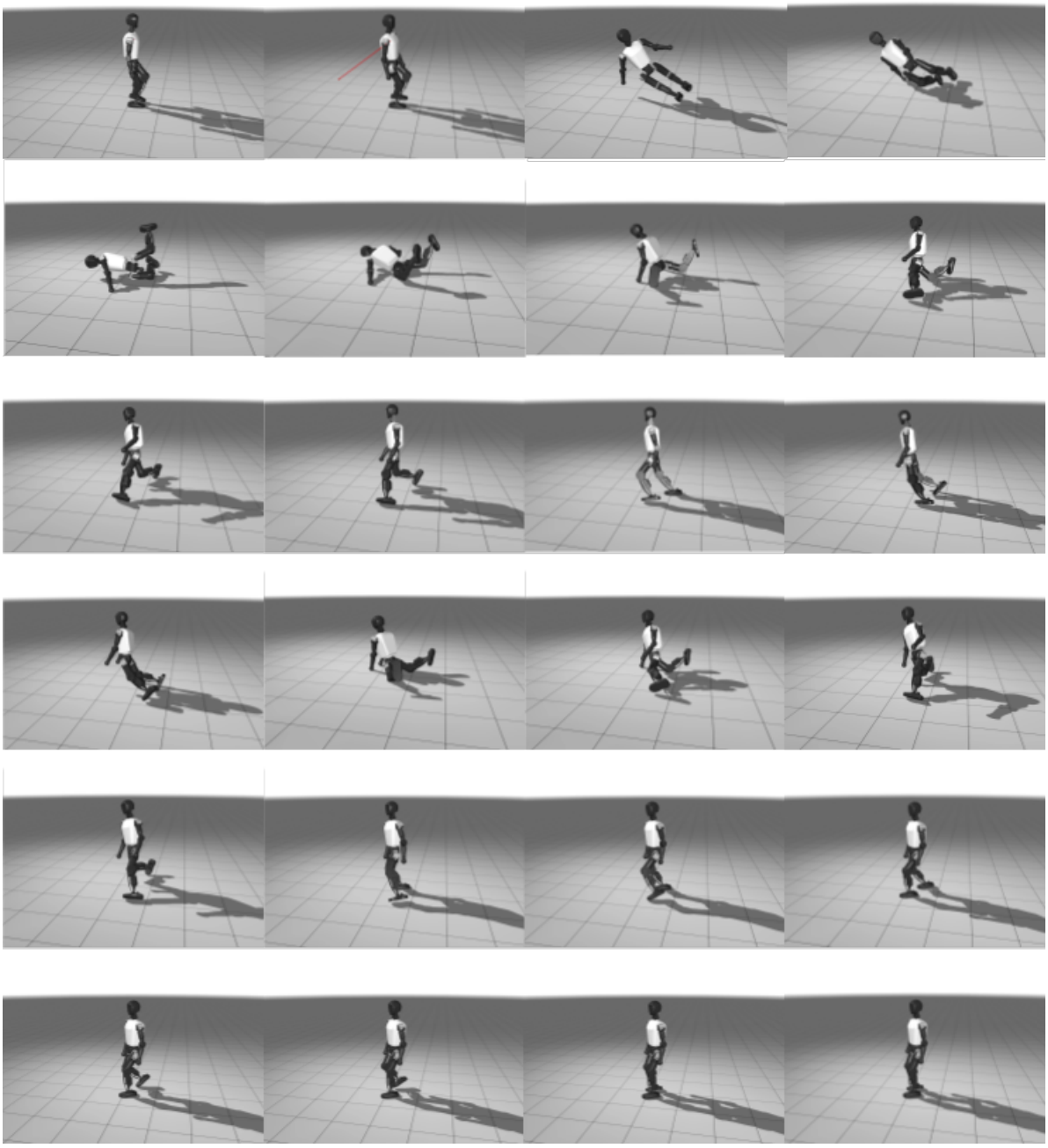

Figure D.21: Baseline backwards push evaluation 10